НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
Факультет інформатики та обчислювальної техніки
Кафедра технічної кібернетики

«На правах рукопису»
УДК 004.8

До захисту допущено:

Завідувач кафедри
_____ Ігор ПАРХОМЕЙ
(підпис)
“___”_____2020 р.

Магістерська дисертація

на здобуття ступеня магістра

за освітньо-професійною програмою «Інформаційне забезпечення роботехнічних систем»

зі спеціальності 126 «Інформаційні системи та технології»

на тему: «Багатокритеріальна система оптимізації налаштування нейронних мереж»

Виконав:
студент ІІ курсу, групи ІК-91мп
Любаченко Микола Олександрович _____

Керівник:
професор, д.т.н., доц.,
Чумаченко Олена Іллівна _____

Консультант з нормоконтролю:
доцент, к.т.н., доц.,
Пасько Віктор Петрович _____

Рецензент:
ст.викладач, к.т.н.
Василенко Микола Павлович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки
Кафедра технічної кібернетики

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Ігор ПАРХОМЕЙ
(підпис)

«___»_____2020 р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Любаченкові Миколі Олександровичу

1. Тема дисертації Багатокритеріальна система оптимізації налаштування нейронних мереж, науковий керівник дисертації Чумаченко Олена Іллівна, д.т.н., доцент, затверджені наказом по університету від «26» жовтня 2020 р. № 3133-с

2. Термін подання студентом дисертації _____

3. Об’єкт дослідження – штучні нейронні мережі

4. Предмет дослідження багатокритеріальні генетичні алгоритми, що будуть налаштовувати нейронні мережі

5. Перелік завдань, які потрібно розробити – аналіз проблематики налаштування нейронних мереж, аналіз алгоритмів для вирішення проблеми, розробка алгоритму багатокритеріальної системи налаштування нейронних мереж.

6. Орієнтовний перелік ілюстративного матеріалу – шість плакатів

7. Орієнтовний перелік публікацій – 1

8. Консультанти розділів дисертації

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
НК	Пасько В.П., доцент		
Перевірка на співпадіння	Лісовиченко О.І., доцент		

9. Дата видачі завдання _____

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Формування проблематики		
2	Аналіз існуючих алгоритмів		
3	Постановка задачі		
4	Розробка структурної схеми		
5	Розробка алгоритму		
6	Розробка ПЗ		
7	Тестування ПЗ		
8	Практичне застосування ПЗ		
9	Попередній захист		
10	Нормоконтроль		
11	Перевірка на співпадіння		
12	Захист		

Студент Микола ЛЮБАЧЕНКО

Науковий керівник Олена ЧУМАЧЕНКО

АНОТАЦІЯ

Штучні нейронні мережі є об’єктом дослідження, а предметом дослідження розглядаються ті генетичні методи та алгоритми нейронних мереж, які будуть налаштовувати нейронні мережі.

Робота присвячена розробці програмного засобу, що допомагає в підборі параметрів для структурно-параметричного синтезу штучних нейронних мереж, та дозволяє отримати нейромережеві моделі, придатні для виробничого застосування. У дипломі розглянуті сучасні підходи до вирішення поставленого завдання. На їх основі розроблено архітектуру програмного застосунку, який являє собою програмну систему, яка складається з окремих компонентів, що робить її гнучкою для змін, легко керованою та дає можливість інтеграції з іншими системами. Застосунок побудований із використанням сучасних програмних засобів, може бути розгорнутий на широкому спектрі обчислювальних систем, є платформонезалежним.

Отримані результати можуть бути корисними при застосуванні у багатьох галузях, що пов’язані з класифікації, апроксимацією,

Ключові слова: штучні нейронні мережі, багатокритеріальна оптимізація, налаштування нейронних мереж, застосування штучних нейронних мереж, генетичні алгоритми.

Розмір пояснювальної записки становить 88 аркушів, містить 29 ілюстрацій, 26 таблиць, 6 додатків.

SUMMARY

Neural networks are the object of research, and the subject of research is the evolutionary methods and algorithms of neural networks, the parameters of which will be offered to the user.

The work is devoted to the development of software that helps in the selection of parameters for structural and parametric synthesis of artificial neural networks, and allows to obtain neural network models suitable for industrial use. The diploma considers modern approaches to solving the problem. Based on them, a software application architecture was developed, which is a software system that consists of individual components, which makes it flexible for change, easy to manage and allows integration with other systems. The application is built using modern software, can be deployed on a wide range of computer systems, is platform-independent.

The obtained results can be useful in many areas related to robotic systems and system that are related to classification, approximation, prediction.

Keywords: artificial neural networks, multiobjective optimization, neural network settings, application of artificial neural networks, genetic algorithms.

Explanationary note size – 88 pages, contains 29 illustrations, 26 tables, 6 applications.

Пояснювальна записка

до магістерської дисертації

на тему: *Багатокритеріальна система оптимізації налаштування нейронних мереж*

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ШНМ – штучна нейронна мережа.

БД – база даних.

ПЗ – програмне забезпечення.

СППР – система підтримки прийняття рішень.

БО – багатокритеріальна оптимізація.

ШІ – штучний інтелект.

ГА – генетичний алгоритм

ВСТУП

Сучасні системи, які вимагають рішення проблем високої складності все більше і більше використовують штучні нейронні мережі. До таких проблем можна віднести вирішення задачі класифікації, прогнозування, кластеризації, прийняття рішень, апроксимування, аналізу даних, оптимізації. Використання ШНМ можна широко зустріти в багатьох галузях – будівництво (наприклад, знаходження оптимальних параметрів конструкції), медицина (розпізнавання хвороб), економіка (прогнозування курсу валют) та ін. Але при використанні ШНМ виникає проблема прийняття рішень щодо типу нейронної мережі та її параметрів в залежності від багатьох критеріїв.

На сьогодні відомо, що питання настроювання параметрів нейронних мереж розглядається в роботах Синєглазова В.М., Miikkulainen R, Liang J, Meyerson E, Rawal A, Fink D, Francon O, Raju B, Shahrzad H, Navruzyan A, Duffy N, Hodjat B, Olson.

Наукова новизна :

Алгоритм визначення найкращого генетичного алгоритму для навчання нейронної мережі з визначенням її структури (кількість шарів, кількість нейронів в шарах) та значень вагових коефіцієнтів, який відрізняється від відомих тим, що з метою покращення якості навчання поставлено та розв’язано задачу оптимізації, яка відшукує найкращі з алгоритмів.

В результаті ми отримуємо систему, яка повинна збільшити ефективність вирішення проблем нейронних мереж за рахунок багатокритеріальної системи оптимізації – оптимальність параметрів дозволить зменшити складність навчання нейронних мереж при отриманні оптимальної точності.

Це може використовуватися на практиці для покращення вирішення задач з використанням нейронних мереж – в основному буде використовуватися для зменшення використання ресурсів при вирішенні проблем.

РОЗДІЛ 1. ШТУЧНИЙ ІНТЕЛЕКТ ЯК ЗАСІБ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ ОБРОБКИ ВЕЛИКИХ МАСИВІВ ІНФОРМАЦІЇ

1.1. Необхідність використання штучного інтелекту в системах обробки інформації

На теперішній час штучний інтелект використовується у багатьох сферах. Штучний інтелект (ШІ) дозволяє машинам вчитися на досвіді, пристосовуватися до нових входів і виконувати завдання, що виконуються людьми. Більшість прикладів, які ми можемо сьогодні почути – комп'ютери, що грають в шахи, машини, які їздять без водія, що значною мірою покладається на глибоке вивчення та обробку природної мови. Використовуючи ці технології, комп'ютери можуть навчати виконувати конкретні завдання, обробляючи великі обсяги даних та розпізнаючи структури даних. Отже, сферою використання ШІ є системи обробки інформації.

Враховуючи, що системи обробки інформації є сукупністю технічних засобів, програмного забезпечення, а також методів обробки інформації для здійснення автоматичної обробки інформації, використання ШІ є необхідним. Основна мотивація та необхідність використання штучного інтелекту – здатність вирішувати складні проблеми (велика кількість змінних, складність алгоритмів), тобто такі завдання класифікації,апроксимації, прогнозування, прийняття рішень, управління є ідеальними.

1.2. Нейронні мережі та їх класифікація

Нейронні мережі – це набір алгоритмів, розроблених відповідно до людського мозку, які призначені для розпізнавання закономірностей. Вони інтерпретують сенсорні дані через своєрідне машинне сприйняття, групуючи вихідні дані. Шаблони, які вони розпізнають, є числовими, містяться у векторах, в які мають бути перекладені всі реальні дані, будь то зображення, звук, текст чи часові ряди. Нейронні мережі використовуються в різних завданнях, найпоширенішими є задачі прогнозування, прийняття рішень, розпізнавання образів, оптимізації, кластеризації, класифікації.

Також як нейронні мережі можна описати послідовність нейронів, з'єднаних між собою синапсами (зв'язками). Звідси ви можете розглянути основні компоненти ШНМ.

Основна складова ШНМ – штучний нейрон або просто нейрон – обчислювальний блок, який отримує певну інформацію, виконує на ній певні прості обчислення та передає її далі. Існує три основних типи нейронів – вхідні, вихідні, приховані. Однак нейронні мережі формуються з досить великої кількості нейронів – які утворюють "шари". Шар – розташування нейронів. Відповідно, загалом існує вхідний рівень, який приймає інформацію; *n* схованих шарів для обробки інформації, де *n* – кількість шарів; зв'язок вихідний шар – вихідний рівень. Кожен нейрон має два параметри – вхід і вихід або вхід, вихід. Вихід – це загальна інформація всіх нейронів попереднього шару (якщо це вхідний нейрон, то вхід дорівнює виходу), що нормується за допомогою функції активації *f(x)* і передається у вихідне поле. Нормалізація дозволяє перетворювати дані в інтервалі від 0 до 1 або ж від -1 до 1 у цьому діапазоні і оперувати нейронами.

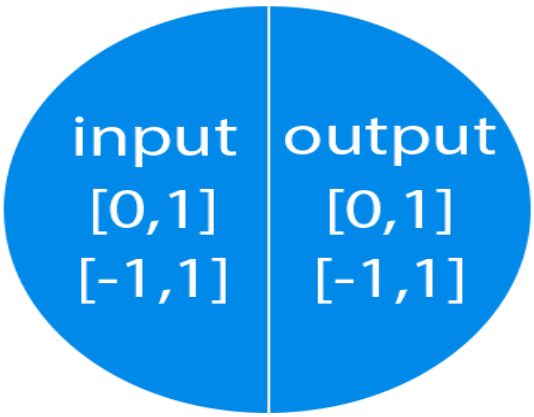


Рисунок 1.1. – Межі оперування нейрону

Інший компонент – синапс або зв'язок між двома нейронами. Основним параметром будь-якого з'єднання є вага, він дозволяє змінювати вхідну інформацію, яка передається від одного нейрона до іншого. Найпростіший приклад – домінування інформації. Якщо ми маємо 2 нейрони і відповідні 2 ваги, то домінуватиме інформація нейрона, який має найбільшу вагу. Якщо припустити, що вага – *w*, *x*, *y* – входи та виходи, то базова нейронна мережа показана на рис. 1.2. – одношаровий перцептрон.

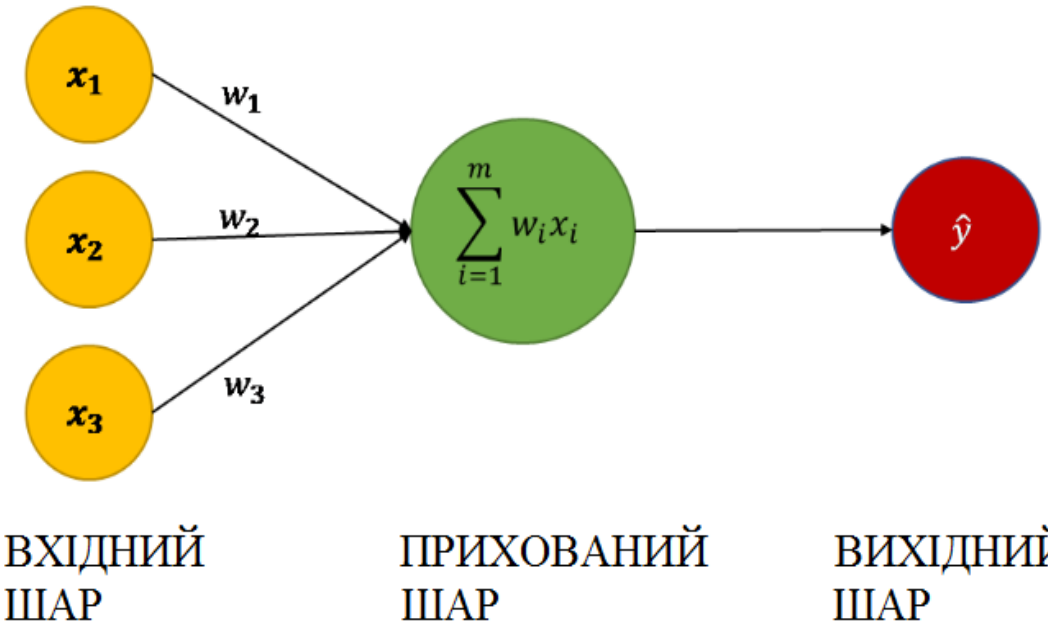


Рисунок 1.2. – Архітектура одношарового перцептрона

Загалом, нейронні мережі класифікуються на 2 типи за типом з'єднань: мережі прямого розподілу та мережі зворотного зв'язку (періодичні).

Штучні нейронні мережі, що мають явний напрямок, порядок шарів (вхідний, прихований, вихідний) називаються нейронними мережами прямого поширення. До них належать одношарові багатошарові перцептрони. І рекурентні – мережі з певними циклами, які добре підходять для обробки послідовностей. Сфера повторюваних мереж – обробка інформації (текст, відео, зображення, емоції, прогнозування чогось на основі попередніх даних).

Ви також можете класифікувати нейронні мережі за такими ознаками, як кількість шарів, алгоритм навчання, тип завдання (прогнозування, класифікація, оптимізація, кластеризація тощо). За кількістю шарів – одношарові, багатошарові. За алгоритмом навчання – з викладачем, безвикладача, з підкріпленням.

Прикладами нейронних мереж по їх структурі є:

– неглибокі нейронні мережі (англ. Shallow neural networks);
– глибокі нейронні мережі (у тому числі, багатошаровий перцептрон);
– згорткові нейронні мережі (Convolutional neural networks – CNN);
– рекурентні нейронні мережі (Recurrent neural networks – RNN)
– LSTM мережі або довга короткочасна пам’ять (Long Short Term Memory);
– уваго орієнтовані нейронні мережі (Attention based neural networks);

– генеративно-змагальні нейронні мережі (англ. Generative Adversarial Networks – GAN).

Неглибокі нейронні мережі мають єдиний прихований шар перцептрона. Одним із поширених прикладів дрібних нейронних мереж є колаборативне фільтрування [53]. Прихований шар персептрону буде навчений представляти подібність між сутностями для того, щоб формувати рекомендації. Система рекомендацій у Netflix, Amazon, YouTube тощо використовує версію колаборативної фільтрації, щоб рекомендувати свої продукти відповідно до інтересів користувачів

Нейронні мережі з декількома прихованими рівнями називаються глибокими нейронними мережами. Усі наступні нейронні мережі є формою глибокої нейронної мережі, доопрацьованої для вирішення проблем, що стосуються конкретних сфер. Загалом, вони допомагають нам досягти універсальності. Враховуючи достатню кількість прихованих шарів нейрона, глибока нейронна мережа може наблизитись, тобто вирішити будь-яку складну реальну проблему. Теорема універсальної апроксимації є ядром глибоких нейронних мереж для навчання та підгонки будь-якої моделі. Кожна версія глибокої нейронної мережі розробляється повністю зв'язаним шаром максимально об'єднаного продукту множення матриць, який оптимізований алгоритмами зворотного розповсюдження. Ми будемо продовжувати вивчати вдосконалення, що призводять до різних форм глибоких нейронних мереж. На рис. 1.3. показано структуру глибоких нейронних мереж.

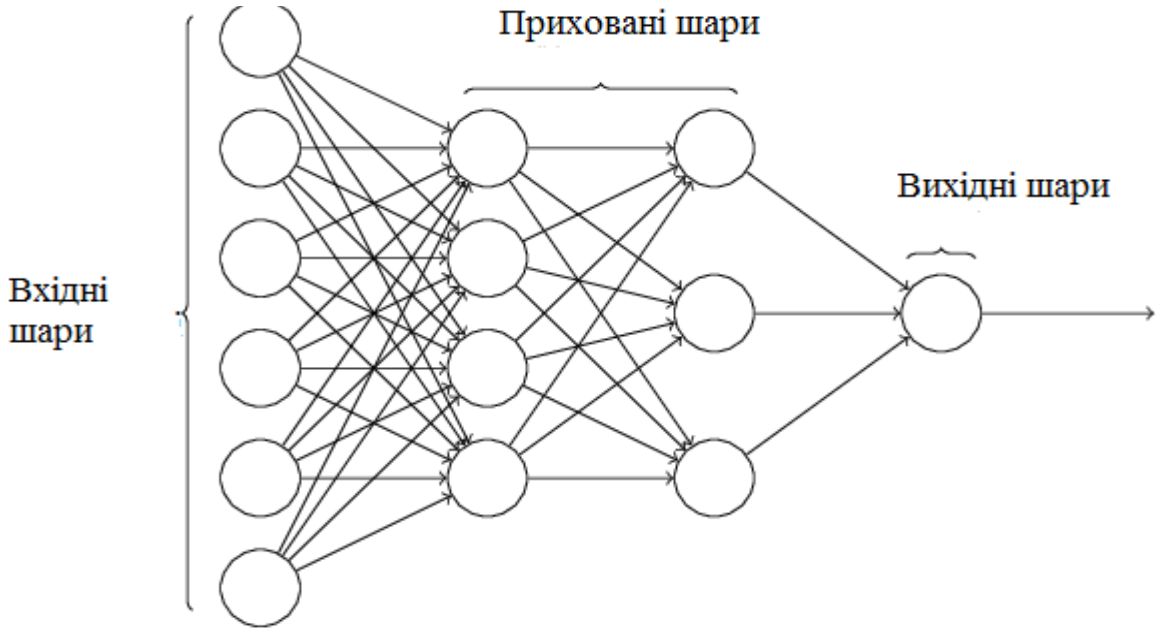


Рисунок 1.3 – Глибокі нейронні мережі

CNN є найзрілішою формою глибоких нейронних мереж для отримання найточніших, тобто кращих, ніж результати людини в комп'ютерному зорі. CNN складаються із шарів Convolutions, створених скануванням кожного пікселя зображень у наборі даних. Оскільки дані отримують апроксимацію шар за шаром, CNN починає розпізнавати візерунки і тим самим розпізнавати об’єкти на зображеннях. Ці об'єкти широко використовуються в різних додатках для ідентифікації, класифікації тощо. Нещодавні практики, такі як навчання передачі в CNN, призвели до значного поліпшення неточності моделей. Google Translator та Google Lens – це найбільш сучасні приклади CNN.

RNN – це найновіша форма глибоких нейронних мереж для вирішення проблем у НЛП. Простіше кажучи, RNN подають вихідних даних декількох прихованих шарів назад на вхідний шар для агрегування та переносять наближення до наступної ітерації (епохи) вхідного набору даних. Це також допомагає моделі самостійно вчитися та швидше коригує передбачення. Існують різні варіанти RNN, такі як довгострокова короткочасна пам'ять (LSTM), керований рекурентний блок (GRU), тощо. На рис. 1.4. нижче активація з h1 та h2 подається на вхід x2 та x3 відповідно.

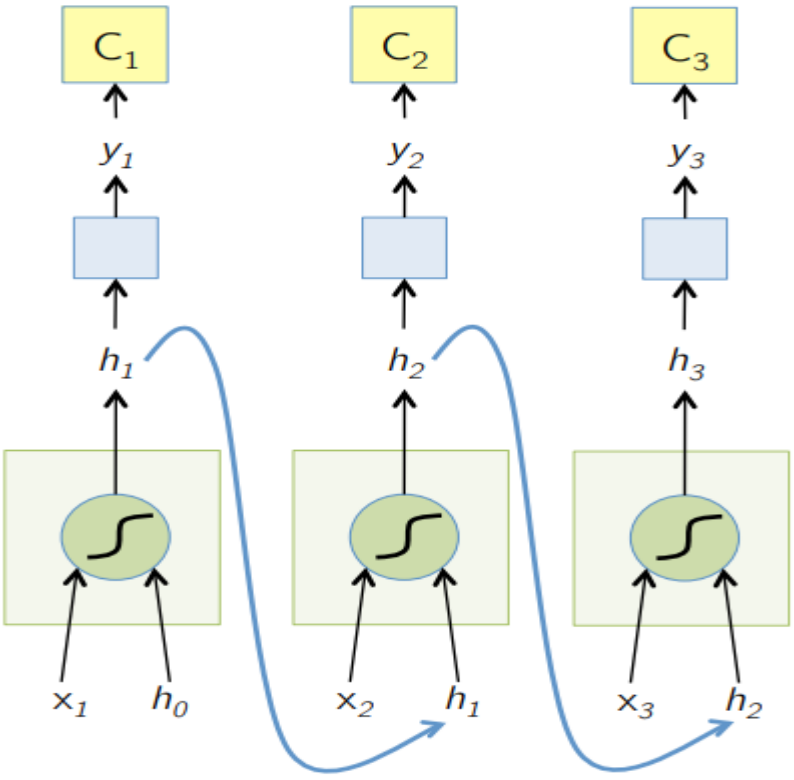


Рисунок 1.4. – RNN

LSTM розроблені спеціально для вирішення проблеми зникаючих градієнтів з RNN. Зникаючі градієнти трапляються з великими нейронними мережами, де градієнти функцій втрат мають тенденцію наближатися до нуля, роблячи паузу нейронних мереж для вивчення. LSTM вирішує цю проблему, запобігаючи функціям активації в її періодичних компонентах і шляхом іммутування збережених значень. Ця невелика зміна дала значні вдосконалення остаточній моделі, в результаті чого технічні гіганти адаптували LSTM у своїх рішеннях.

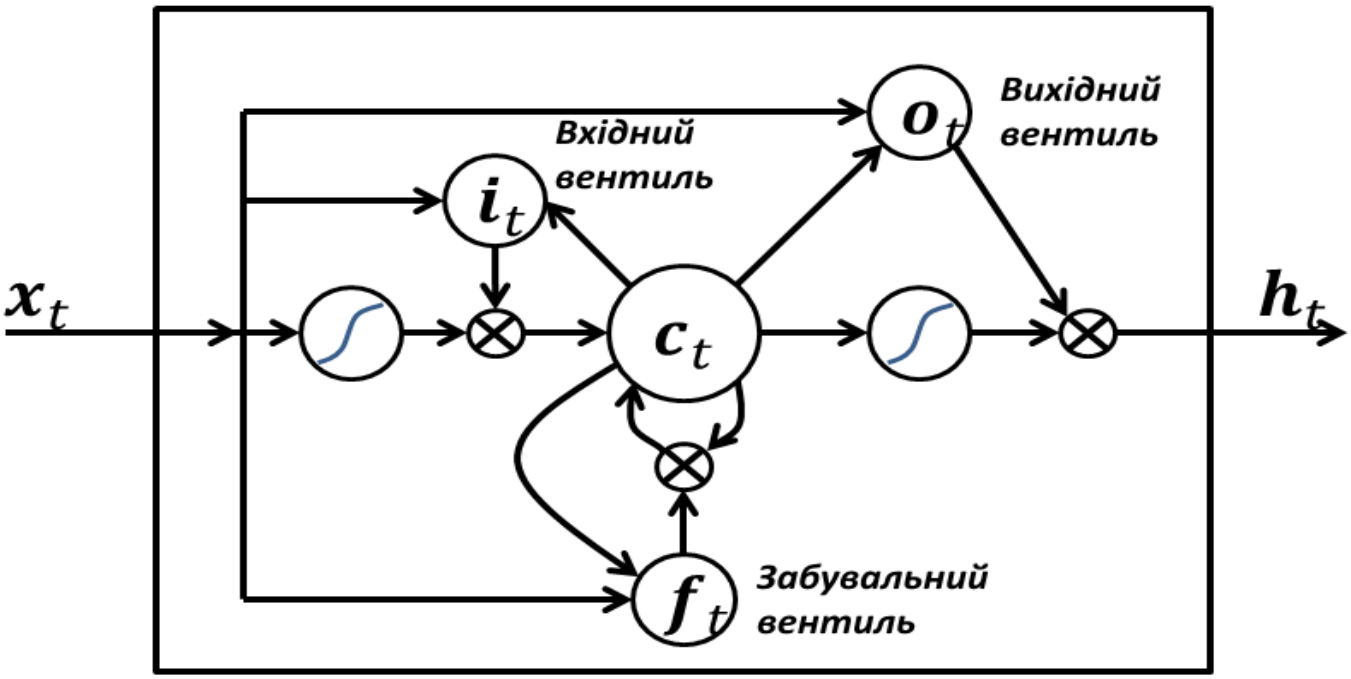


Рисунок 1.5. – Структура LSTM

Уваго орієнтовані нейронні мереж (*англ.* Attention based neural networks) поволі завойовують навіть нові RNN на практиці. Ці ШНМбудуються, зосереджуючись на частині підмножини інформації, яку вони отримують, усуваючи тим самим величезну кількість фонової інформації, яка не потрібна для поточного завдання. Кілька моделей уваги, складених ієрархічно, називається трансформатором. Ці трансформатори ефективніше паралельно запускати стеки, щоб вони давали сучасні результати із порівняно меншими даними та часом на навчання моделі. Розподіл уваги стає дуже потужним при використанні з CNN / RNN і може створювати текстовий опис до зображення. Використання уваго орієнтованих нейронних мереж показано на рис. 1.6.

Рисунок 1.6. – Використання уваго орієнтованих нейронних мереж

Суть ідеї генеративно-змагальної нейронної мережі в комбінації двох нейромереж, при якій одночасно працює два алгоритми "генератор" і "дискримінатор". Завдання генератора – створювати образи заданої категорії. Завдання дискриминатора – намагатися розпізнати створений образ. Використанням цих ШНМ є генерація реального світу в іграх, генерація рухів, яка особа не робила, генерація нових абсолютно реальних зображень. На рис. 1.7. показано генерацію зебри на основі коня.

Рисунок 1.7. – Генерація зебри на основі коня

Один із найбільш повних переліків з їх структурою (в тому числі й тих, що були описані зверху) показано в додатку Г. Необхідно зазначити, що структура не завжди показує як проходить процес навчання, тому рекомендується розібратися з навчанням кожної нейронної мережі потім [51][52].

1.3. Проблеми навчання нейронних мереж

Навчаючи ШНМ відбувається процес, при якому вільні параметри ШНМ конфігуруються шляхом моделювання середовища, в яке цей ШНМ вбудований. Викладання нейронної мережі означає, що ми повідомили її про вимоги. Процес навчання можна описати на рис. 1.8.

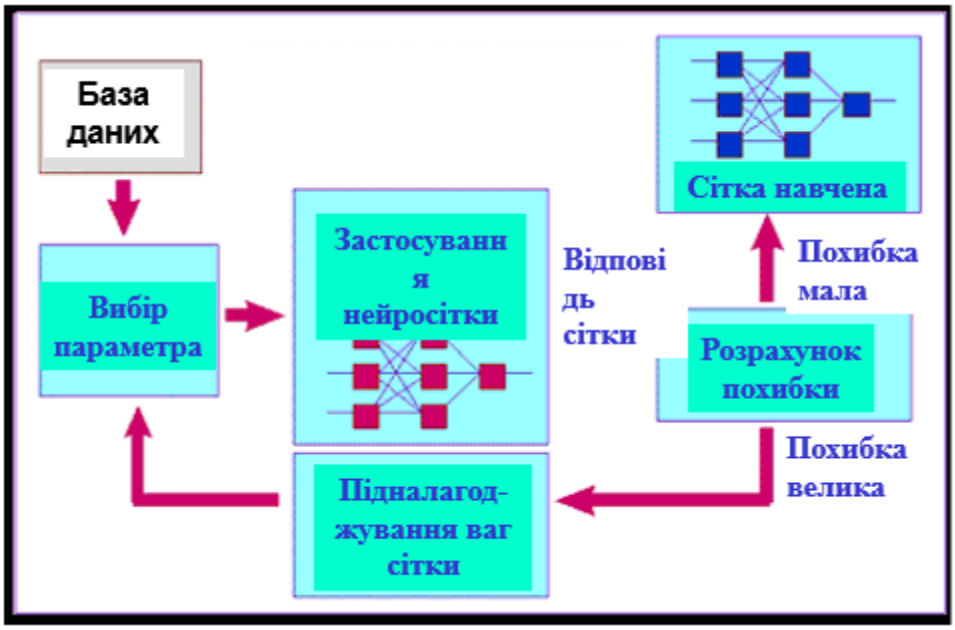


Рисунок 1.8. – Процес навчання ШНМ

Для навчання потрібно виконати такі дії:

1. Вибрати правильну структуру нейронної мережі.
2. Правильно вибрати параметри навчання: крок навчання, кількість параметрів для навчання, кількість прикладів для навчання, алгоритм навчання.
3. Підготувати вхідні дані.

Основною проблемою, яка виникає під час навчання, є перенавчання. Цей процес відбувається у випадку дуже тривалого навчання, недостатньої кількості тематичних досліджень або дуже складної архітектури ШНМ. Одним із способів боротьби є розподіл навчальної вибірки на два набори – тренувальний і тестований. Навчання відбувається на навчальному наборі, на наборі тестів – побудована модель перевіряється. Перенавчання можна спостерігати, змінюючи помилки в цих двох вибірках. Перенавчання – це процес коли вірогідність помилки навчання на об’єктах тестової вибірки виходить явно вищою за середню помилку на навчальній вибірці. Процес закінчення реального навчання та початку перепідготовки зображено на рис. 1.9.

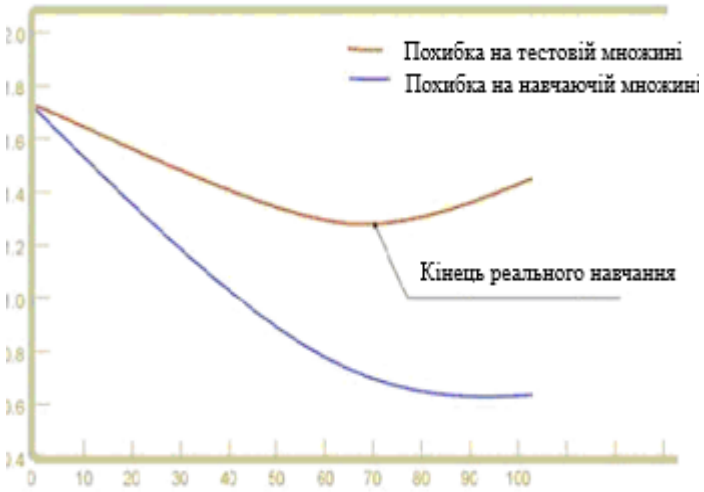


Рисунок 1.9. – Процес завершення реального навчання та початок перенавчання

Метод, показаний на рис. 1.9. називається методом раннього навчання. Для боротьби з перенавчанням також може бути використаний алгоритм проріджування зв’язків та конструктивні методи.

Алгоритм витончення зв’язків – це техніка, яка дозволяє зменшити різноманітність можливих конфігурацій навчених нейронних мереж з мінімальною втратою їх апроксимуючих можливостей.

Для цього замість дзвоноподібної форми апріорної функції розподілу ваги, яка характерна для звичайного тренування, коли ваги «поширюються» від початку, використовується тренувальний алгоритм, в якому функція розподілу ваги зосереджена в основному в "нелінійній” області порівняно великих значень ваг. Викуп малих значень ваг у цій техніці показано на рис. 1.10.

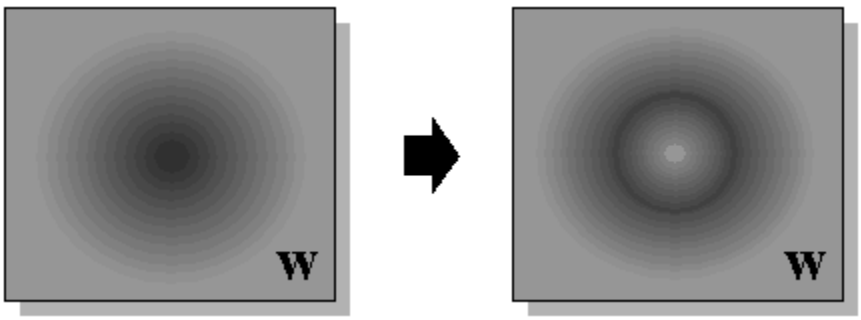


Рисунок 1.10. – Погашення малих значень ваги методом витончення з'єднань

Конструктивні алгоритми характеризуються збільшенням швидкості навчання завдяки тому, що складність навчання пропорційна квадрату числа ваг, а навчання «частинами» вигідніше, ніж вивчення цілого. Формула 1.1. показує це.

$$w^2 = (\sum_k w_k)^2 > \sum_k {w_k}^2. \quad (1.1.)$$

Каскадно-кореляційний метод навчання нейронних мереж показаний на рис. 1.11.:

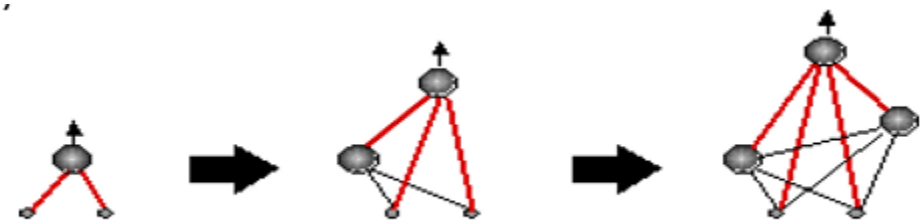


Рисунок 1.11. – Каскадно-кореляційна методика навчання нейромереж: додавання проміжних нейронів з фіксованими вагами (тонкі лінії)

До інших проблеми при вивченні ШНМ включають специфічні алгоритми, відсутність конкретних методів для створення більш оптимальної моделі, шум у даних, дуже великий набір даних, неправильну кількість нейронів. Ці проблеми дуже складні і потребують вирішення в майбутньому – збільшенням обчислювальної потужності неможливо вирішити ці проблеми.

1.4. Методи навчання нейронних мереж

При обговоренні методів навчання нейронних мереж, ми маємо на увазі викладання з викладачем, без викладача або з підкріпленням.

Навчаючись у викладача, ШНМ вивчає конкретний спостережуваний набір даних і передбачає відповіді, які використовуються для оцінки точності алгоритму на навчальних даних. Під час навчання без викладача модель використовує немарковані дані, з яких алгоритм самостійно намагається витягти особливості та залежності.

Викладання з підкріпленням – це щось середнє між двома методами. Він використовує невелику кількість маркованих даних та великий набір немаркованих даних. Посилене навчання навчає алгоритм системою стимулів. Агент отримує зворотний зв’язок у вигляді винагороди за правильні дії (як при дресируванні тварин).

Для кожного методу навчання розглянемо приклади відповідних даних та завдань.

Навчання з викладачем вимагає повного набору позначених навчальних даних для навчання моделі на всіх етапах. Цей комплект означає, що кожен приклад у навчальному наборі відповідає відповіді, яку повинен отримати алгоритм. Відповідно до цього методу навчання, якщо набір даних був позначений у навчальному наборі, мережа базується на цьому порівнянні прикладів навчального набору. В основному, навчання з учителем використовується для вирішення двох типів задач: класифікації, регресії.

Вирішуючи проблему класифікації, алгоритм передбачає дискретні значення, які відповідають певній кількості класів, до яких належать об’єкти. У навчальному наборі даних із фотографіями тварин кожне зображення матиме відповідну мітку – "кішка", "ведмідь" або "черепаха". Якість алгоритму оцінюється тим, наскільки точно він може правильно класифікувати нові фотографії з ведмедями та черепахами. Але проблема регресії пов'язана з безперервними даними. Один приклад, лінійна регресія, обчислює очікуване значення змінної у з урахуванням конкретних значень x. Як приклад, лінійна регресія обчислює очікуване значення змінної у з урахуванням конкретних значень x.

Більш утилітарні завдання машинного навчання включають велику кількість змінних. Наприклад, нейронна мережа, яка може прогнозувати ціну квартири в Сан-Франциско на основі її площі, місцезнаходження та доступності громадського транспорту. Алгоритм виконує роботу експерта, який розраховує ціну квартири на основі тих самих даних.

Таким чином, навчання з викладачем є найбільш підходящим для виконання завдань, коли існує досить великий набір даних (надійних) для вивчення алгоритму. Але це не завжди так, ідеально позначені дані отримати непросто, тому перед алгоритмом часто стоїть завдання знайти невідомі раніше відповіді – тоді вам потрібне навчання без викладача.

При навчанні без викладача модель має набір даних, але не має чітких вказівок щодо того, що з цим робити. Нейронна мережа намагається самостійно знаходити співвідношення в даних, витягуючи корисні функції та аналізуючи їх.

Залежно від завдання, модель систематизує дані по-різному. Основні завдання для навчання без викладача – кластеризація, пошук аномалій, асоціацій, автокодерів. Суть кластеризації полягає в тому, що алгоритм повинен знаходити подібні дані, знаходити спільні ознаки та групувати їх разом. Виявлення аномалій. Наприклад, коли відбувається шахрайство з банківськими картками – наприклад, коли одна і та ж картка використовується в США та Данії в один день. Тобто для виявлення цих аномалій у даних використовується навчання без викладача.

Асоціації – це коли на основі одних характеристик об'єкта модель може передбачити інші, з якими існує зв'язок. Це використовується в інтернет-магазинах для пропонування інших подібних товарів – разом із телефоном, вам пропонують купити зарядний пристрій, навушники тощо. Автоекодери використовуються для отримання вхідних даних, їх кодування та відтворення вихідних даних з отриманого коду. У реальних ситуаціях, якщо ви додаєте шумні та оригінальні версії зображень для навчання, автоенкодери дозволяють видаляти шум із відеоданих, зображень, медичних сканувань, щоб поліпшити якість даних.

Але при навчанні без викладача важко визначити точність моделі – оскільки дані не мають правильних відповідей або позначень. Але навіть при всьому цьому ви можете отримати хороші дані, знаходячи залежності.

Часткова підготовка викладача використовує як марковані, так і немарковані дані для викладання, що дає цьому методу велику перевагу. Це застосовується, коли важко отримати всі дані, наприклад, у медицині відмічають кілька даних, а потім застосовують цей метод. Також сьогодні дуже популярний генеративний метод навчання для створення навчальної вибірки, що імітує меншу навчальну вибірку. Для цього методу потрібні дві мережі – генератор, дискримінатор. Генератор імітує навчальну вибірку, і дискримінатор перевіряє ці дані, щоб перевірити, чи справжні вони.

Відеоігри засновані на системі стимулів. Завершіть рівень і отримайте винагороду. Переможіть всіх монстрів і заробіть бонус. Потрапили в пастку – кінець гри, не потрапляйте. Ці заохочення допомагають гравцям зрозуміти, як найкраще діяти в наступному раунді гри. Без зворотного зв'язку люди просто приймали випадкові рішення і сподівалися перейти на наступний рівень гри.

Посилене навчання діє за тим же принципом. Відеоігри – це популярне тестове середовище для дослідів. Агенти штучного інтелекту намагаються знайти найкращий спосіб досягти мети або поліпшити ефективність для певного середовища. Коли агент штучного інтелекту діє, щоб допомогти досягти певної мети, він отримує винагороду. Передбачення цих кроків є глобальною метою отримати в кінцевому підсумку максимальну винагороду.

Приймаючи рішення, агент штучного інтелекту освоює відгуки, нові стратегії та рішення, які можуть призвести до більших вигод. Цей підхід використовує довгострокову стратегію. Якщо говорити про приклад шахів, то один наступний хід не завжди допомагає в підсумку перемогти. Тому агент намагається максимізувати загальну винагороду.

Це ітераційний процес. Чим більше рівнів має зворотній зв'язок, тим краща тактика агента штучного інтелекту. Цей процес особливо корисний для навчальних роботів, які керують запасами на складі або автономних транспортних засобах. Так само, як учні школи, кожен алгоритм навчається по-різному. Але завдяки різноманітності доступних методів питання полягає у тому, щоб вибрати правильний і навчити свою нейронну мережу розуміти навколишнє середовище.

Висновки до розділу

У цьому розділі описується необхідність використання штучного інтелекту і особливо – штучних нейронних мереж, їх класифікацію, структури багатьох ШНМ, проблеми, методи навчання.

РОЗДІЛ 2. СТРУКТУРНО-ПАРАМЕТИЧНИЙ СИНТЕЗ НЕЙРОННИХ МЕРЕЖ З ВИКОРИСТАННЯМ ГЕНЕТИЧНИХ АЛГОРИТМІВ

2.1. Постановка задачі

Структурно-параметричний синтез – це процес, в результаті якого визначається структура об’єкта та знаходять параметричні значенні складових його елементів, так що виконуються умови завдання синтезу. Якщо синтезований об'єкт є оптимальним за будь-яким критерієм (критеріями), то і сам синтез є оптимальним.

Використані математичні та комп’ютерні моделі, що застосовуються для автоматизації структурно-параметричного синтезу об’єктів, суттєво відрізняються від тих моделей, що застосовують при автоматизації параметричного синтезу. Звідси випливає, що якщо при параметричному синтезі структуру об’єкта в синтезальному процесі не змінюється, то при виконанні процесу структурно-параметричного синтезу відбувається зміна параметрів об’єкта та зміна структури.

Постановка задачі структурно-параметричного синтезу в нашому випадку визначить структуру моделі та їх параметри два критерії (складність і точність).

Для постановлення задачі структурно-параметричного синтезу в нашому випадку необхідно визначити модельну структуру та її параметри, а саме – модельну складність нейронної мережі та точність.

Нехай дано максимальну кількість нейронів A в нейронній мережі, що будується для апроксимації залежності від вибірки вихідних даних $<X, Y>$, де $X = \{X_i\}$ – набір значень характеристик (ознакк), які описують об'єкт або процес; $Y = \{y_p\}$ – масив значень параметра на виході в даній вибірці; $X_i = \{X_{ip}\}$ – це i-та ознака у вибірці, $i = 1, 2, \ldots, L$; x_{ip} – значення i-го атрибута для p-го зразка вибірки, $p = 1,2,\ldots, M$; y_p – значення передбачуваного параметра для p-го екземпляра; L – загальна кількість об’єктів в оригінальному наборі; m – кількість зразків.

Тоді завданням нейромодельного структурно-параметричного синтезу є знаходження моделі виду $HC=HC(S, W, B)$, для яких $\xi(HC, X, Y) \rightarrow \min$, при цьому $S=S(L,A)$ – матриця, що визначає наявність синаптичних зв’язків між елементами мережі (вхідні функції, нейрони); $W=W(S)$ – матриця ваг, що відповідає співвідношенням, присутнім у мережі; $B = B(S)$ – вектор зсуву мережевих нейронів; $\xi(HC,X,Y)$ – критерії, що визначають ефективність моделі нейронної мережі HC для

апроксимування співвідношення між набором параметрів на вході – *X* і відповідним вектором значень параметрів на виході –*Y*.

2.2. Генетичні алгоритми та їх особливості

Під методами ми маємо на увазі навчальні алгоритми – процедуру, яка використовує правила навчання для встановлення ваг. У цій роботі ми розглянемо генетичні алгоритми як найпопулярніші та сучасні методи навчання ШНМ.

2.2.1. Фізичні основи генетичних алгоритмів

Генетичними алгоритмами називаються методи вирішення, що базуються на природному відборі – процесі, що керує еволюцією в біології. Ці методи використовуються як для обмежених так і для необмежених оптимізаційних задач. Генетичний алгоритм неодноразово модифікує індивідуальні рішення. На кожному кроці генетичний алгоритм випадковим чином відбирає індивідуумів із поточної популяції (сукупності індивідуумів) для батьків і використовує їх для розмноження дітей для наступного покоління. Протягом декількох наступних поколінь популяція «еволюціонує» до оптимального рішення. Застосування генетичних алгоритмів є величезним для різноманітних оптимізаційних задач, які не підходять для стандартних оптимізаційних алгоритмів. До цього можна включати завдання з розривними цільовими функціями, недиференційованими, стохастичними або дуже нелінійними. Задачі цілочисельного програмування можуть бути вирішеними з використанням ГА, де деякі складові можуть бути цілими числами.

Основні етапи будь-якого генетичного алгоритму показано на рис. 2.1.

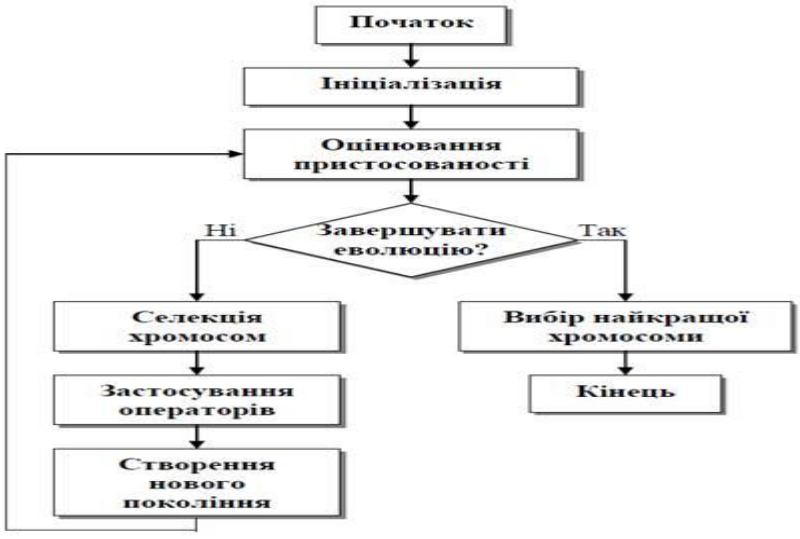


Рисунок 2.1. – Основні етапи будь-якого генетичного алгоритму

До переваг ГА можна додати простоту у розумінні, широка застосованість в практичних задачах (де потрібно щось оптимізувати, оскільки ці методи можуть вирішувати нелінійні проблеми), паралелізм та рішення.

Серед переваг генетичних алгоритмів можна назвати: легкість розуміння, велике практичне значення (завдяки тому, що ці алгоритми можуть широко використовуватися в будь-якій задачі, де щось потрібно оптимізувати, і ці методи можуть вирішувати нелінійні проблеми), паралелізм, рішення.

Звичайно, в генетичних алгоритмах наявні недоліки. Налаштування генетичних алгоритмів для складних реальних проблем не є явним. Під кожну конкретну проблему важливо вабрати та кодувати потенційні рішення.

2.2.2. Оператори генетичних алгоритмів

Базові кроки будь-якого ГА: формулювання базисної популяції, розрахунок пристосованості, селекція (популяційний відбір індивідуумів), оператор кросинговеру та/або мутації, розрахунок пристосованості для всіх індивідуумів та створення нового покоління. Необхідно зазначити, що кроки повторюються.

Звідси випливають важливі генетичні проблеми:

– як розраховувати чисельність популяції;
– що таке пристосованість;
– оператор селекції;
– оператор кросинговеру;
– оператор мутації.

2.2.2.1. Визначення чисельності популяції

У загальному випадку чисельність популяції залежить від кількості генів. Дев'ять генів мають 16 хромосом, 16 генів мають 32 хромосоми. Доцільно розрахувати чисельність популяції залежно від генів. Оптимальний розмір популяції буде в 2 рази перевищувати кількість генів, але не більше 100. Слід зазначити, що невелика популяція швидко сходиться, але алгоритм можна захопити в локальному екстремумі. Крім того, із збільшенням чисельності населення основна увага приділяється пошуку рішень у конкретних регіонах, ігноруючи інші.

2.2.2.2. Визначення пристосованості

Пристосованість (або фітнес-функція) оцінює, наскільки дане рішення наближене до оптимального рішення бажаної проблеми. Це визначає, наскільки підходить рішення.

У генетичних алгоритмах кожне рішення зазвичай представлене як рядок двійкових чисел, відоме як хромосома. Ми повинні протестувати ці рішення і придумати найкращий набір рішень для вирішення даної проблеми. Отже, кожному рішенню потрібно присуджувати оцінку, щоб вказати, наскільки воно наблизилось до відповідності загальній специфікації бажаного рішення. Цей бал генерується шляхом застосування функції фітнесу до тесту або результатів, отриманих з тестованого рішення.

Будь-яка пристосованість повинна задовольняти наступним вимогам:

- пристосованість повинна бути чітко визначена. Читач повинен мати можливість чітко розуміти, як обчислюється бал придатності.
- пристосованість слід виконувати ефективно. Якщо вона стане слабким місцем алгоритму, то загальна ефективність генетичного алгоритму буде знижена.
- пристосованість повинна кількісно вимірювати, наскільки відповідне вирішення проблеми вирішується.
- пристосованість повинна давати інтуїтивні результати. Найкращі/гірші кандидати повинні мати найкращі/гірші оцінки.

Не існує жорсткого правила про те, що певна функція повинна використовуватися в певній задачі. Однак певні функції були прийняті вченими[54] з питань даних щодо певних типів проблем.

Як правило, для класифікаційних завдань, де використовується навчання під контролем, такі показники помилок, як евклідова відстань[54] та відстань Манхетена[54], широко використовуються як пристосованість.

Для задач оптимізації основні функції, такі як сума набору обчислених параметрів, що відносяться до області проблем, можуть бути використані як пристосованість.

Для прикладу нехай ми врахуємо набір з 5 генів, які можуть містити одне з двійкових значень 0 і 1, ми повинні придумати послідовність, що має всі 1. Тож ми повинні максимально збільшити кількість 1s. Це можна розглядати як проблему оптимізації. Отже, пристосованість розглядається як кількість 1s, присутніх у геномі. Якщо є п'ять одиниць, то це має максимальну підготовленість і вирішує нашу проблему. Якщо немає 1s, то пристосованість має мінімальну пристосованість.

Основними генетичним операторами є оператор селекції, оператор кросинговеру, оператор мутації.

2.2.2.3. Оператор селекції

Мало уваги приділяється оператору селекції. Без цього оператора генетичні алгоритми – це лише прості випадкові методи, які кожного разу дають різні значення. Індивідууми, які мають вищу пристосованість, мають велику ймовірність бути обраними наступним поколінням цим оператором.

Оператор селекції – це найважливіший параметр, який може вплинути на результативність роботи ГА. Він спрямований на використання найкращих характеристик рішень-кандидатів з метою вдосконалення цих рішень протягом поколінь, що, в принципі, повинно спрямовувати ГА на сходинку до прийнятного та задовільного вирішення актуальної проблеми оптимізації.

Найпоширенішими техніками селекції[55] є:

- рулеткова селекція (*англ.* Roulette Wheel Selection – RWS);
- стохастична універсальна селекція (*англ.* Stochastic Universal Sampling – SUS);
- лінійно рангова селекція (*англ.* Linear Rank Selection – LRS);
- експонціально рангова селекція (*англ.* Exponential Rank Selection – ERS);
- турнірна селекція (*англ.* Tournament Selection – TOS);
- селекція усіканням (*англ.* Truncation selection – TRS).

2.2.2.4. Оператор кросинговеру

За імітування процесу індивідуумного схрещування відповідає оператор кросинговеру. Представимо, що ми маємо 2 індивідуума з хромосомами $X = \{x_i, i = 1, L\}$ і $Y = \{y_i, i = 1, L\}$. В середині хромосоми ми випадково створюємо точку, яка розкладає батьківські хромосоми на дві частини та відбувається обмін

ними. Називається ця точка – точкою зупинки або точкою перетину (точка кросинговеру). Описаний процес показано на рис. 2.2.

Рисунок 2.2. – Процес кросинговеру

Існує декілька видів точкових кросинговерів. Єдиноточковим можна назвати той, який розрізає хромосоми батьків в одній випадковій точці. Але існують також n-точкові оператори кросинговеру. Якщо дві точки зупинки, то кросинговер є 2-точковим. Якщо n точок зупинки, то він є n-точковим.

Окрім окреслених типів кросинговерів, залишається також однорідний кросинговер. Його особливість полягає в тому, що значення кожного біта в хромосомі сукупності обчислюється випадково з відповідних батьківських бітів. Нехай існує деяке значення 0 <par_0<1, і якщо випадкове число більше par_0, то n-те положення першого потомства є n-м бітом першого батьківського, а n-те положення другого – n-го біт другого батька. Інакше перше потомство одержує біт другого батька, а друге – першого. Цей обмін проводиться для всіх бітів хромосоми. Вірогідність кросинговеру найвища серед генетичних операторів і зазвичай становить 60% і більше.

2.2.2.5. Оператор мутації

Оператор мутації потрібен, щоб "вибити" популяцію з місцевого екстремуму і допомагає захистити від передчасного наближення. Це досягається за допомогою обертання значення бітів у хромосомах, тобто процесу інвертації, що й показано на рис. 2.3.

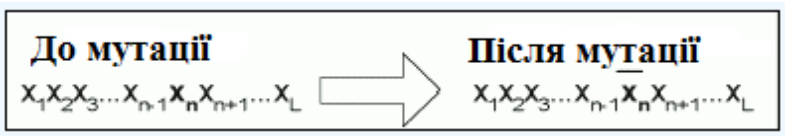


Рисунок 2.3. – Процес мутації шляхом інвертування випадкового біта у хромосомі

Оператор мутації визначається не тільки в випадково визначеній точці. Можна задавати кількість хромосомних точок для інвертування також. Інвертування підряд

підходить чудово також. Вірогідність мутації є значно меншою за вірогідність виконання оператору кросинговеру, більше 1% є досить рідким явищем. Рекомендованими варіантами є 1/*length* та 1/*size*, якщо вважати, що величина популяції – size, а *length* – довжина хромосоми.

Слід також зазначити, що існує думка, що оператор мутації є основним оператором пошуку та відомими алгоритмами, які не використовують оператори, крім мутації. Слід також зазначити, що нам потрібно оцінити вірогідність виконання операцій кросинговеру та мутації.

2.3. Класифікація генетичних алгоритмів

На сьогоднішній день існує багато генетичних алгоритмів, але найпопулярнішими є VEGA (векторний генетичний алгоритм), FFGA (багатоцільовий генетичний алгоритм Фонсеки та Флемінга) або MOGA (також званий багатоцільовим генетичним алгоритмом), NPGA (генетичний алгоритм Ніше-Парето), SPEA та SPEA2 (еволюційний алгоритм сили Парето), QGA (квантовий генетичний алгоритм).

2.3.1. VEGA

Девід Шаффер (1984) [16-18] розширив програму ГЕНЕЗИС Грефенштетта [Grefenstette 1984], включивши багатокритеріальні функції. Підхід Шаффера полягав у використанні розширення простого генетичного алгоритму (*англ*. Simple genetic algorithm – SGA), яке він назвав векторним генетичним алгоритмом (VEGA), і яке відрізнялося від SGA лише селекцією. Цей оператор був модифікований таким чином, що при кожному поколінні генерувались ряд підгруп, здійснюючи пропорційну селекцію відповідно до кожного критерію. Таким чином, для проблеми з критеріями «k» генеруються підгрупи k розміром N/k (припускаючи загальний розмір популяції N). Ці підгрупи буде перемішано, щоб отримати нову популяцію розміром N, на яку застосовуватиметься ГА, в якому оператори кросинговеру та мутації використовуються звичайним способом. На рис. 2.4. показано структурне представлення цього процесу.

Головною перевагою цього алгоритму є його простота, тобто такий підхід досить легко реалізувати. Річардсон та співавтори. (1989) [19] зазначають, що перетасовування та злиття всіх підгруп відповідає усередненню пристосованих

компонентів, пов'язаних з кожним із критеріїв. Оскільки Шаффер використовував пропорційне призначення пристосованості, ці компоненти пристосованості, в свою чергу, були пропорційні самим критеріям. Отже, отримана очікувана пристосованість відповідає лінійному поєднанню цілей, де ваги залежать від розподілу популяції в кожному поколінні, як показали Річардсон та співавтори.

Основним недоліком цього є те, що коли ми маємо увігнуту поверхню компромісу, певні точки в увігнутих регіонах не будуть знайдені за допомогою цієї процедури оптимізації, в якій ми використовуємо лише лінійну комбінацію критеріїв, і було доведено, що це істина незалежно від набору ваг, що використовуються. Отже, головною слабкістю цього підходу є його нездатність виробляти Парето-оптимальні рішення за наявності неопуклих просторів пошуку.

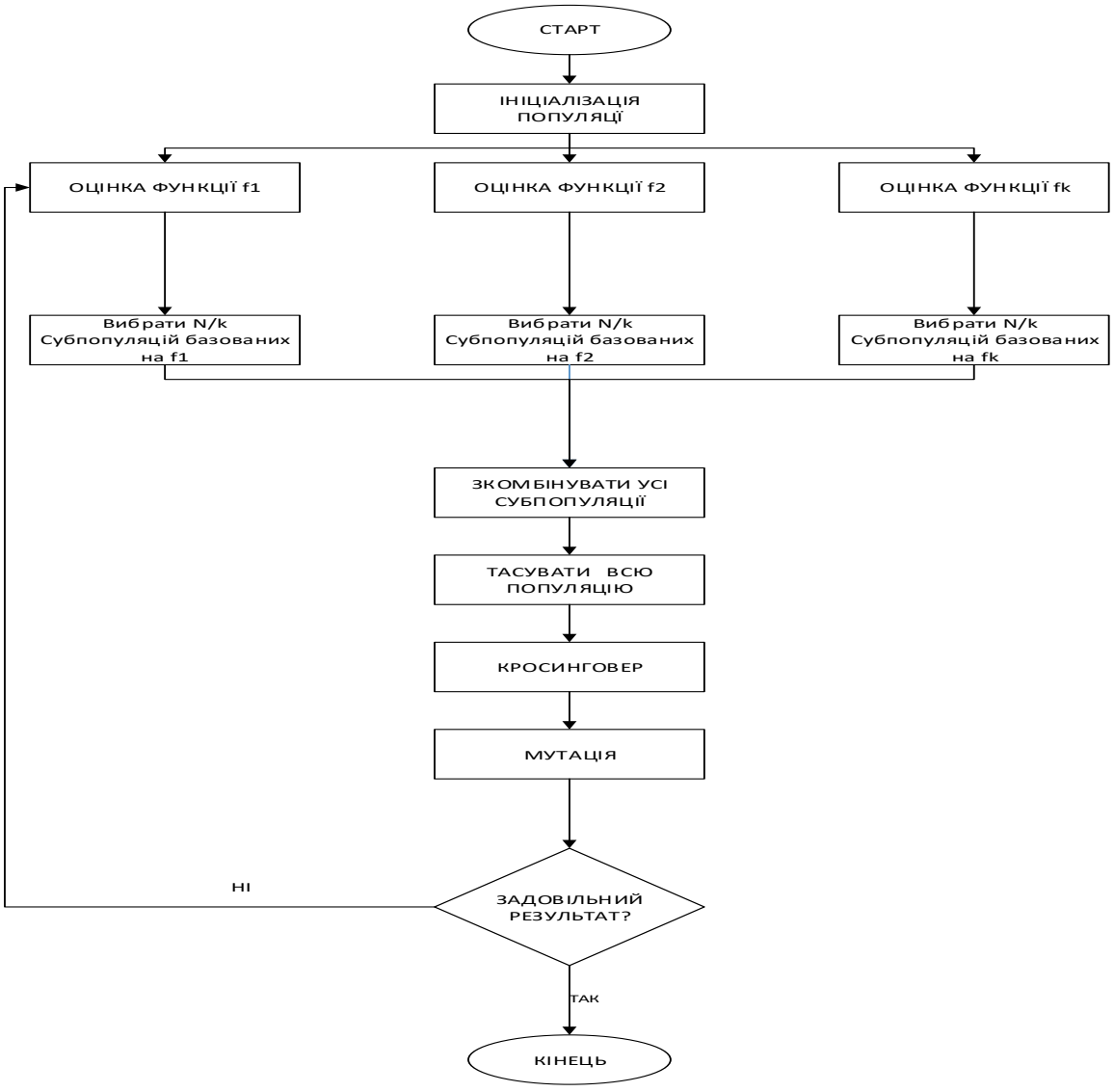


Рисунок 2.4. – Алгоритм VEGA

2.3.2. FFGA

Фонсека та Флемінг (1993) [20-22] реалізували пропозицію Голдберга по-іншому. Спершу обговоримо, що пропонує Голдберг щодо рейтингу Парето.

Голдберг запропонував схему ранжирування Парето в (1989), де рішення x при генерації t має відповідний цільовий вектор x_u, а n – розмір популяції, ранг рішення визначається наступним алгоритмом.

2.3.2.1. Алгоритм FFGA

1. curr_rank = 1; m = n;
2. Поки $n! = 0$ робити
3. Для $i = 1\ldots, m$ робити
4. Якщо x_u is недомінований по rank(x, t)=curr_rank кінець робити
5. Для i=1,…,m
6. робити
7. Якщо rank (x, t)= curr_rank Видалити x з популяції n=n-1;
8. кінець робити
9. curr_rank = curr_rank +1 m = n
10. кінець для поки

Це означає, що в рейтингу Парето перевіряється вся популяція, і всім індивідуумам, які не домінують, присвоюють ранг «1». Потім видаляють цих індивідуумів із популяції, які мають ранг «1». Знову виявляють усіх недомінованих індивідуумів від решти популяції та присвоюють ранг «2». Таким чином процедура продовжується, поки всі рішення не отримають необхідний ранг.

Але в багатокритеріальних генетичних алгоритмах перевіряється вся популяція, і всі індивідуумам, яким не домінують, присвоюють ранг «1». Інші особи класифікуються, перевіряючи їх домінування щодо решти популяції наступним чином.

Наприклад, індивідуум x_i у поколінні t, в якій переважають $p_i^{(t)}$ p (t) особини в поточному поколінні. Його поточне становище в рангу індивідуумів може бути дано за

$$Rank(x_i, t) = 1 + p_i^{(t)} \quad (2.1.)$$

Після завершення процедури ранжування настав час призначити пристосованість кожному індивідууму. Фонсека та Флемінг запропонували два методи визначення пристосованості:

– ранг базоване визначення пристосованості;
– методи формування ніш.

Призначення фітнесу на основі рангу виконується наступним чином:

– Відсортувати популяцію за рангом;
– Призначити пристосованість, інтерполюючи від найкращого (ранг «1») до гіршого (ранг $n \leq N$) звичайним способом, відповідно до певної функції, зазвичай лінійної, але не обов'язково;
– Середньо оцінити рівень пристосованості індивідуумів з однаковим рангом, щоб усі вони брали участь з однаковою швидкістю. Ця процедура підтримує постійну пристосованість глобальної популяції, зберігаючи відповідний тиск селекції, як визначено використовуваною функцією.

Як зазначали Голдберг та Деб, такий тип заблокованої пристосованості, швидше за все, призведе до великого тиску підбору, що може спричинити передчасну збіжність. Щоб уникнути цього, Фонсека та Флемінг використовують другий метод (тобто метод формування ніш) для розподілу популяції по оптимальній Парето-області, але замість того, щоб здійснювати обмін значеннями параметрів, вони використовували обмін значеннями цільової функції.

Головною перевагою є те, що ефективно і порівняно просто у впровадженні. Ефективність цього методу сильно залежить від коефіцієнта розподілу (σ_{share}). Однак Фонсека та Флемінг розробили хорошу методологію для обчислення такої величини для свого підходу.

2.3.3. NPGA

Хорн, Нафплоїтіс і Голдберг [23] [24] запропонували NPGA на основі турніру домінування Парето та обміну класами еквівалентності.

Парето-домінувальний турнір. В основному це схема турнірної селекції, заснована на Парето-домінуванні. У цій схемі селекційний набір порівняння, що складається з певної кількості (tdom) індивідуумів, вибирається випадковим чином

серед популяції на початку кожного процесу селекції. Два індивідууми для селекції вибираються навмання серед популяції. Потім кожного з індивідуумів порівнюють з кожним індивідуумом у наборі порівняння. Якщо в одному переважає набір порівняння, а в іншому ні, пізніший відбирається для відтворення. Якщо в жодному або обох не переважає набір порівняння, тоді ми переходимо до другої техніки.

Обмін класами еквівалентності. Оскільки обидва індивідууми однакові, тобто або домінновані, або не домінуючі, цілком ймовірно, що вони знаходяться в одному класі еквівалентності. Тож у цьому випадку ми вибираємо “найкраще підходящих” за наступною процедурою.

Ми вибираємо радіус ніші (σ_{share}), і відповідно до цього радіусу кандидати, які мають найменшу кількість індивідуумів у популяції, є найбільш підходящими. На наступному рис. 2.4. показано, як працює ця процедура: тут ми максимізуємо вздовж осі x і мінімізуємо по осі y.

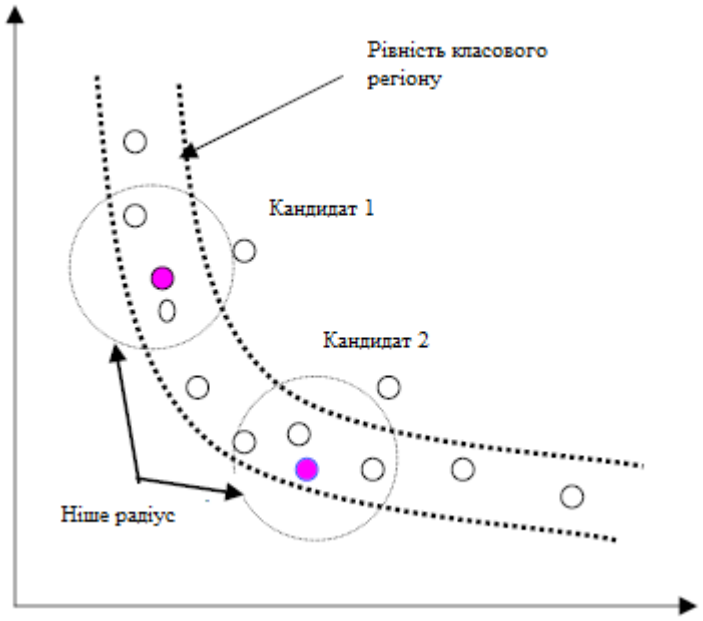


Рисунок 2.4. – Обмін класами еквівалентності

У цьому випадку набір кандидатів для відбору не переважає набір порівняння. Отже, за кількістю ніш це показує, що кандидат 1 найкраще підходить. Тут t_{dom} вибирається лише один раз для певного покоління t. Після створення нової популяції застосовують генетичного оператора, подібного до інших методів.

2.3.3.1. Алгоритм NPGA

1. Ініціалізація чисельності популяції n.
2. Вибір t_{dom} розмір хромосоми випадково з поточної популяції.

3. Випадкий підбір двох хромосом з поточної популяції.

4. Вибір n особин, спираючись на наступну процедуру: Порівняти дві хромосоми з t_{dom} на предмет недомінування за попереднім визначенням. Якщо одна є домінованою, а і інша не є домінованою – вибрати ту, що не є домінована. Якщо обидва не домінують або домінують, то методом формування ніш виберіть найкращу хромосому (індивідуум).

5. Застосувати кросинговер та мутацію, щоб отримати нову популяцію.

6. Перевірити, чи задовольняються критерії ефективності, якщо ні, то перейдіть до кроку 2, інакше до кроку 7.

7. Зупинка.

Оскільки цей підхід не застосовує Парето-селекцію до всієї популяції, а лише до її сегменту при кожному прогоні, його головні переваги полягають у тому, що він дуже швидкий і що він створює хороші непереважаючі фронти, які можна утримувати для великої кількості поколінь.

Ефективність цього методу сильно залежить від коефіцієнта розподілу (σ_{share}) та хорошої турнірної чисельності (t_{dom}) і є складним для реалізації.

2.3.4. NSGA

NSGA відрізняється від простих ГА лише у тому вигляді, в якому використовується оператор селекції. Оператор кросинговер та домінувальні рішення також важливі для того, щоб отримати хороший розподіл рішень в оптимальному Парето-фронті. Пристосованість виконується у два етапи.

1. Присвоєння однакової фіктивної пристосованості всім рішенням певного рівня домінування.

2. Застосовування стратегії обміну.

Тепер ми обговоримо деталі цих двох етапів:

По-перше, всім рішенням на першому непереважаючому фронті присвоюється пристосованість, рівна чисельності популяції. Це стає максимальною пристосованістю, яку може мати будь-яке рішення для будь-якої популяції. На основі стратегії спільного використання, якщо рішення має багато сусідніх рішень на одному фронті, його фіктивна пристосованість зменшується на коефіцієнт і обчислюється загальна пристосованість. Фактор залежить від кількості та близькості

сусідніх рішень. Після того, як усім рішенням на першому фронті присвоєні значення пристосованості, визначається найменше загальне значення пристосованості.

Після цього особам, які перебувають на другому рівні домінування, присвоюється фіктивна пристосованість, рівна кількості, меншій за найменшу загальну пристосованість попереднього фронту. Це гарантує, що жодне рішення на другому фронті не має загальної пристосованості, ніж будь-яке рішення на першому фронті. Це підтримує тиск на рішення, щоб вести до оптимального Парето-фронту. Метод спільного використання знову використовується серед осіб другого фронту, і виявляється загальна пристосованість кожної людини. Ця процедура продовжується до тих пір, поки всі особи не отримають загальну пристосованість.

Після методу призначення пристосованості використовується рулеткова селекція (RWS) для селекції N осіб. Після цього застосовують кросинговер та мутацію. Спільна пристосованість обчислюється наступним чином:

Враховуючи набір рішень n_1 у l-му непереважаючому фронті, кожен з яких має фіктивне значення f_1, спільна процедура виконується наступним чином для кожного рішення $i = 1,2,3, \dots, n_1$.

1. Обчислити нормовану міру евклідової відстані з іншим розв'язком j у l-му непереважаючому фронті, як показано нижче:

$$d_{ij} = \sqrt{\sum_{p=1}^{p} \left(\frac{x_p^{(i)} - x_p^{(j)}}{x_p^{(u)} - x_p^{(s)}}\right)^2} \qquad (2.3.)$$

де P є числом критеріїв в проблемі. Ці параметри $x_p^{(u)}$and $x_p^{(s)}$є нижніми та верхніми границями x_p.

2. Відстань d_{ij} порівнюється із заздалегідь заданим параметром σ_{share} та обчислюється таке значення функції спільного використання:

$$sh(d_{ij}) = \begin{cases} 1 - \left(\frac{d_{ij}}{\sigma_{share}}\right)^2, \text{якщо } d_{ij} \le \sigma_{share} \\ 0, \text{інакше} \end{cases} \qquad (2.5.)$$

3. Інкрементування j. Якщо$j \le n_1$, перейти до кроку 1 та обчислити $sh(d_{ij})$. Якщо $j > n_1$, обчислити кількість ніш для і-го рішення таким чином:

$$m_i = \sum_{j=1}^{n_1} sh(d_{ij}) \quad (2.6.)$$

4. Погіршити фіктивну пристосованість f_1 i-го рішення в l-му фронті, що не домінує, для обчислення загальної пристосованості $\acute{f}_i$, наступним чином:

$$\acute{f}_i = \frac{f_1}{m_i} \quad (2.7.)$$

Ця процедура продовжується для всіх $i = 1,2,3 \ldots, n_1$ та знайденої відповідно $\acute{f}_i$. Після цього для подальшої обробки знайдено найменше значення $\acute{f}_i^{\min}$ з усіх $\acute{f}_i$ l-го непереважаючого фронту. Фіктивна пристосованість наступного непереважаючого фронту визначається$f_{l+1} = f_i^{min} - \varepsilon_1$ де ε_1 – невелике додатне число.

Вищевказана процедура спільного використання вимагає заздалегідь заданого параметра σ_{share}, який можна обчислити наступним чином:

$$\sigma_{share} \approx 0.5/\sqrt{q}^{1/p} \quad (2.8.)$$

де q – бажане число різних Парето-оптимальних рішень. Хоча розрахунок σ_{share} залежить від цього параметра q, було показано, що q≈10 добре працює для багатьох тестових задач.

Головною перевагою цього методу є те, що він може обробляти будь-яку кількість критеріїв, і це робить розподіл у просторі значень параметрів замість простору ціннісних значень об'єкта, що забезпечує кращий розподіл людей і дозволяє отримати кілька рівноцінних рішень.

Деякі дослідники зазначають, що це неефективно, якщо врахувати як обчислювальну ефективність вироблених Парето-фронтів. Ще одним недоліком є те, що він більш чутливий до σ_{share}.

2.3.5. SPEA, SPEA-2

Загальна схема SPEA (Еволюційний алгоритм сили Парето) [6]:

- ініціалізація: генерування 1-ої сукупності та створюється порожнього зовнішнього Парето-оптимального набору (архіву);
- оновлення Парето-оптимального набору. Якщо величина Парето-оптимального набору перевищує зазначений ліміт, подальші мережі Парето знищуються кластеризаційним методом. Для зменшення Парето-

набору до контрольованого розміру використовується середній алгоритм ієрархічної кластеризації на основі з'єднань. Він виконує ітераційне поєднання сусідніх кластерів для досягнення необхідної кількості груп;

- обчислення значення функції пристосованості для зовнішнього Парето-оптимального набору та сукупності індивідуумів;
- Відбір за допомогою турірної селекції: популяція та зовнішній набір об’єднуються, і будь-які дві особи вибираються випадково. Що стосується їхньої функції пристосованості то найкращий з них рухається до пулу (*англ.* mating pool). Пул – це сукупність популяцій, що перетинаються, які проходять операції мутації та кросинговеру, щоб створити нову популяцію;
- нову популяцію продукують операції мутації та кросинговеру;
- установка $t=t+1$. Якщо критерій зупинки ($t>T$) не виконаний, перейдіть до кроку 2; в іншому випадку члени архіву представляються як оптимальний набір Парето.

SPEA значно відрізняється від своїх попередників, оскільки:

- концепція Парето-домінування використовується для присвоєння скалярної придатності особам;
- особи, які не домінують над іншими представниками популяції – індивідуумів, зберігаються окремо у спеціальному зовнішньому наборі (архіві);
- для зменшення кількості осіб, що зберігаються в архіві, здійснюється кластеризація, що, у свою чергу, не впливає на характеристики осіб, набуті в процесі пошуку;

Унікальність та переваги методу SPEA полягає в тому, що:

- він поєднує вищезазначені підходи в одному алгоритмі;
- пристосованість кожної особини популяції визначається лише стосовно осіб зовнішнього архіву, незалежно від того, чи домінують особини популяції одна над одною;

- незважаючи на те, що "найкращі" особи, отримані в попередніх поколіннях, зберігаються у зовнішньому архіві, всі вони беруть участь у відборі;
- для запобігання передчасному зближенню використовується спеціальний механізм формування ніш, де розподіл загальної придатності здійснюється не у значенні відстані між особинами, а на основі домінування Парето.

Одним з недоліків SPEA можна визнати те, що більша частина ресурсів та часу витрачається на процедуру кластеризації, яка забезпечує підтримку різноманітності популяцій.

При розробці SPEA2 основною метою було усунення потенційних недоліків попередника (SPEA) та включення останніх результатів, щоб створити потужний та сучасний багатокритеріальний еволюційний алгоритм. Основні відмінності між SPEA2 та SPEA:

- вдосконалена схема призначення функції пристосованості, яка враховує кожного індивіду, скільки осіб домінує над ним і скільки осіб домінує над іншими;
- найближчий метод оцінки щільності сусідів, що дозволяє більш точно керувати процесом пошуку;
- новий метод усічення архівів, який гарантує збереження граничних рішень.

Загалом, одним із найважливіших кроків у SPEA2 є визначення фітнес-функції або пристосованості.

Визначення пристосованості: виконується на основі використання концепції Парето-домінування, алгоритм обчислення якої (пристосованості F) для кожної особини з популяції P_t та архіву A_t має наступну форму.

1) Припустимо, у нас є набір осіб, які складають популяцію Pt та архів B, де кожній особі присвоюється значення $S(i) \in [0,1)$,, називається "сила" (що показує, скільки рішень вона домінує), яка пропорційна кількості членів сукупності $j \in P_t$, для яких $f(i) \geq f(j)$ у разі багатокритеріальної оптимізації. Пропорція така:

$$S(i) = \frac{n}{N+1}, \qquad (2.9.)$$

де N – чисельність популяції, n – кількість особин, які домінують за умови $f(i) \geq f(j)$.

Але "сила" кожної особини та сукупність осіб архіву A_t та сукупність популяцій P_t визначатиметься як сума "сили" на осіб та "сили" з урахуванням домінування, кодованого за критерієм і вище критерію, кодованого *j*:

$$S(i) = |\{j \mid j \in P_t + A_i \wedge i \rangle j\}, \quad (2.10.)$$

де$i \rangle j$ – домінування, кодоване за критерієм і над критерієм, кодованим *j*.

2) На підставі значення *S*(*i*) обчислюється "груба" (англ. *raw*) величина пристосованості *R*(*i*) особини *i*, яка обчислюється шляхом підсумовування "сил" усіх індивідів j, які домінують або слабо домінують за всіма критеріями:

$$R(i) = \sum_{j \in P_t + A_{t,j>i}} S(j), \quad (2.11.)$$

де P_t– сукупність особин популяції – сукупність осіб архіву A_t.

3) Метод оцінки щільності – це адаптація методу *k*-го найближчого сусіда, де щільність у будь-якій точці є (спадаючою) функцією відстані до *k*-ї найближчої точки даних. Спадною функція називається на деякому інтервалі, якщо для будь-яких значень аргументу з цього інтервалу більше значення аргументу відповідає меншому значенню функції. Інверсія щільності відстані до *k*-го найближчого сусіда береться для оцінки щільності. Для кожного індивіда і відстані (у просторі критеріїв) до осіб *j* в архіві та загальній вибірці підраховуються та зберігаються у списку.

Для обчислення величини пристосованості використовується значення щільності розташування індивідів: для кожного індивіда *i* розраховується декартова відстань від неї до решти індивідів *j* в архіві та сукупності індивідуумів.

Після ранжування списку за зростанням, *k*-й елемент дає бажану відстань до особи та позначається σ_i^k. Це єσ_i^k позначає відстань від індивіда *i* до найближчого *k*-го сусіда. Ми використовуємо k, яке дорівнює квадратному кореню розміру вибірки. Але слід зазначити, що досить часто досить використовувати *k*=1, що призводить до ефективної реалізації.Потім обчислюємо значення щільності *D*(*i*) для окремого *i*:

$$D(i) = \frac{1}{\sigma_i^k + 2}, \quad k = \sqrt{(N + N_A)}, \quad (2.12.)$$

де *N* – це розмір популяції; N_A – чисельність архіву; *k* наближається до найближчого цілого числа.

Дві додаються до знаменника, щоб переконатися, що його значення більше нуля і що $D(i) < 1$.

4) Нарешті, додавши $D(i)$ до початкового значення пристосованості $R(i)$ особин і надає її пристосованості. Отже, остаточне значення функції пристосованості $F(i)$ для індивідуума визначається як $F(i)=R(i)+D(i)$

Час виконання процедури визначення придатності більшою мірою залежить від оцінки щільності – O ($L^2 logL$), тоді як обчислення значень S і R – $O(L^2)$, де $L=M+N$.

Але врешті-решт, основний SPEA-2 складається з наступних етапів:

Біля входу: M, N, T, M – розмір вихідної популяції. N – розмір архіву. T – максимальний розмір поколінь.

Вихід: A * – недомінантний набір.

Крок 1: Ініціалізація: створення початкової сукупності P та порожнього архіву – зовнішнього набору. $A_0 = \emptyset. t = 0.$

Крок 2: Розрахунок пристосованості: Розрахунок значень пристосованості кожного індивіда в P_t та A_t.

Крок 3: Вибір середи: Скопіюйте всіх індивідуумів з P_t і A_t на A_t + 1. Якщо розмір A_{t+1} перевищує N, зменшіть A_t+1 за допомогою оператора відсікання, якщо A_t+1 менше N, заповніть A_t+1 за допомогою домінуючих особин у P_t та A_t.

Крок 4: Завершення: Якщо t більше або дорівнює T, або виконується інший критерій зупинки, тоді множина A* – це набір векторів розв'язків, що представляють недомінантні рішення в A_{t+1}. Кінець.

Крок 5: Відбір: Ми використовуємо двійковий відбір турніру із замінами на A_t $_{+1}$, щоб заповнити басейн .

Крок 6: Варіація: Використовуйте оператори кросинговеру та мутації для пулу та встановлюйте P_{t+1} як результуючу сукупність. Збільште лічильник сукупності (t = t+1) і перейдіть до кроку 2.

2.3.6. QGA

2.3.6.1. Загальний огляд алгоритму

Сучасним генетичним алгоритмом для багатокритеріальної оптимізації є QGA, який інтегрує квантові обчислення з генетичним алгоритмом.

QGA має явні переваги в продуктивності за рахунок введеня квантових понять, як квантовий стан, кванто-обертальний затвір та амплітуда вірогідності в квантових обчисленнях. Представлений метод має малу кількість ітерацій, високу ефективність пошуку та широку застосовуваність. Не рахуючи те, що одна хромосома виражає або може виражати багато станів *i*. Звідси ми робимо висновок про викосу ємність зберігання. Навіть коли кількість індивідів в популяції мізерна, ми можемо казати про відсутність впливу на глобальну оптимізацію.

Таким чином, можливості QGA «впасти» в локальний пошук значно уникнуто. У порівнянні з традиційним генетичним алгоритмом (GA), квантовий генетичний алгоритм не «покладається» на оновлення генів операторами, такими як оператор кросинговеру та мутації, для досягнення еволюції сукупності індивідів. Незважаючи на те, що ці оператори певною міру змінюють амплітуду вірогідності, квантова хромосома уже показує безліч сукупностей індивідів завдяки суперпозиції квантів. А ввід операторів мутації та кросинговеру навіть навпаки зменшує обчислювальну швидкість та продуктивність квантового генетичного алгоритму.

2.3.6.1.1. Квантові біти

Біт – це одиниця інформації у двійковому числі. У традиційному розрахунку існує лише два основних стани, тобто «0» та «1». Після введення квантового поняття бітовий стан стає векторною одиницею у двовимірній комплексній координаті. Окрім 0 та 1, квантовий бітовий стан може бути також лінійною суперпозицією основних станів [33], яка називається станом суперпозиції квантових бітів:

$$|\varphi = \alpha|0 + \beta|1\rangle \, . \qquad (2.13.)$$

Серед них "|•>"– це позначення Дірака для позначення стану. Параметри α і β – ймовірність відповідних станів відповідно. Ймовірність $|0\rangle$ є $|\alpha|^2$, тоді як ймовірність |1> є $|\beta|^2$. Обидві ймовірності задовольняють умові нормування:

$$|\alpha|^2 + |\beta|^2 = 1. \qquad (2.14.)$$

На основі двійкового кодування в генетичному алгоритмі, стан квантового біта $|\phi\rangle$ використовується для кодування цілі та початкового значення. Правила кодування можна виразити наступним чином: за допомогою вираження стану квантової суперпозиції ген може виражати будь-яку квантову бітову інформацію. Крім того,

послідовність геному може бути сформована за складом хромосом. *m*-ну хромосому на схемі можна представити наступним чином:

$$p_m^n = \begin{bmatrix} \alpha_1^n| & |\alpha_2^n| & \dots & |\alpha_{j-1}^n| & \alpha_j^n \\ \beta_1^n| & |\beta_2^n| & \dots & |\beta_{j-1}^n| & \beta_j^n \end{bmatrix}, \quad (2.15.)$$

де *n* – кількість ітерацій, а *j* – квантове число (довжина хромосоми). Повну квантову популяцію, що містить усі сучасні хромосоми, можна виразити як

$$P(t) = \{p_1^t, p_2^t, \dots, p_m^t, \dots, p_T^t\}. \quad (2.16.)$$

2.3.6.1.2. Квантові ворота

Оновлення та еволюція квантової популяції проводяться через квантовий затвор, що є квантовим пристроєм, який може реалізувати логічне перетворення протягом певного інтервалу часу. Квантовий затвор використовується для суперпозиції кванта і призводить до фазової зміни положення гена в хромосомі. Нарешті, ймовірність збігається до 0 або 1 за найкоротший час і досягається оптимальне пошукове рішення. Єдиною вимогою до квантових воріт є

$$U^+ * U = I, \quad (2.17.)$$

де *U* – матричне представлення квантових воріт, U^+ є спряженим транспонуванням, а *I* – матрицею ідентичності. Квантові ворота мають багато форм. У цій конструкції алгоритму метод квантового обертання визначається наступним чином:

$$U(\theta) = \begin{bmatrix} cos\theta_i & -sin\theta_i \\ sin\theta_i & cos\theta_i \end{bmatrix}, \quad (2.18.)$$

де θ – кут повороту. Процес відновлення хромосом можна виразити наступним чином:

$$\begin{bmatrix} \alpha_k^{n+1} \\ \beta_k^{n+1} \end{bmatrix} = \begin{bmatrix} cos\theta_i & -sin\theta_i \\ sin\theta_i & cos\theta_i \end{bmatrix} \begin{bmatrix} \alpha_k^n \\ \beta_k^n \end{bmatrix}, \quad (2.19.)$$

де $[\alpha_k^n, \beta_k^n]^{\mathrm{T}}$ – *K-ий* квантовий біт *N-го* покоління хромосоми:

$$\theta_i = \Delta\theta \times sig, \quad (2.20.)$$

де $\Delta\theta$ визначає швидкість збіжності та точність пошуку алгоритму. Якщо амплітуда $\Delta\theta$ занадто мала, час пошуку збільшується або навіть «застоюється». Якщо амплітуда занадто велика, може статися передчасна конвергенція, і важко отримати оптимальне рішення. sig – знак коефіцієнта кута повороту, а саме напрямку обертання, значення якого визначає напрямок збіжності до оптимального рішення. Коли ibin (поточне

значення індивідуального квантового біта) дорівнює ibbest (оптимальне значення індивідуального квантового біта), sig становить 1/20,5 м; коли ibin та ibbest різні, значення sig наведені в таблиці 2.1.

2.3.6.2. Процес оптимізації QGA

2.3.6.2.1. Кодування та ініціалізація популяції

У квантово-генетичному алгоритмі є три методи кодування.

Таблиця 2.1. Значення sig

ibin	ibbest	*f*(jfval) < *f*(best)	sig			
			$\alpha_i\beta_i > 0$	$\alpha_i\beta_i < 0$	α_i=0	$\beta_i = 0$
0	1	True	1	0	0	± 1
0	1	False	-1	1	± 1	0
1	0	True	-1	1	± 1	0
1	0	False	1	-1	0	± 1

Початковим кодуванням популяції є квантово-бітове кодування, в якому N – довжина кодованих квантових бітів. Псевдокод показаний на рис. 2.5.

Коли спочатку вимірюється популяція, квантовий бітовий код перетворюється в двійковий код, як показано на рис. 2.6. Бінарний код декодується в десятковий при обчисленні адаптованості популяції.

Для оптимізації періоду будівництва та вартості проекту популяцію можна визначити як набір хромосом (рис. 2.7.), що зберігає тривалість усіх послідовних підпунктів, а початкову популяції можна виразити як $P(t = 0) = \{p_1^0, p_2^0 \dots, p_m^0, \dots, p_T^0)$ де T – чисельність популяції. Ймовірності амплітуди популяції $2jT$ становлять $1/2^{0,5}$, що означає, що у початковому стані кожна хромосома знаходиться в лінійному стані суперпозиції з однаковою ймовірністю в $1/2^{0,5}$.

2.3.6.2.2. Оцінка пристосованості

Індивідууми в сукупності індивідуумів можуть бути оцінені на пристосованість. Вища підготовленість (пристосованість) вказує на те, що індивідуум кращий і, швидше за все, виживе. Навпаки, індивідуума з меншою пристосованістю легше усунути. Функція оцінки пристосованості зазвичай узгоджується з цільовою функцією. Оскільки дві протилежні підцілі: час і витрати,

для пошуку мінімальних значень в оптимізаційній моделі рівняння можна змінити наступним чином:

$$Value\ 1 = \frac{1}{C}$$
$$Value\ 2 = \frac{1}{T}. \qquad (2.21.)$$

Значення пристосованості як в десятковому, так і в розрахунковому вимірі було отримано за допомогою недомінантного рішення (інше рішення, яке краще за інше).

2.3.6.3.3. Квантова генетична операція

В квантовій генетичній операції спостерігається $Q(t)$ стан сукупності індивідуумів та порівнюється з існуючим оптимальним рішенням, а потім сукупність з квантово обертовим затвором оновлюється для отримання $Q(t+1)$. Отже отримаємо розрахунок пристосованості. Коли оптимальне рішення в $Q(t+1)$ є кращим, ніж розчин, що зберігається в даний час, збережений розчин замінюється. У процесі оновлення значення сукупності індивідів завжди постійне, при цьому недомінуючі рішення не повторюються.

При виконанні умови припинення виходить набір оптимальних рішень для графіка та вартості підпроцесу. В іншому випадку кроки 2 і 3 повторюються.

```
QGAChrom = zeros(P, N * 2);
for i = 1:P
for j = 1:N * 2
QGAChrom(i, j) = 1/sqrt(2);
end
end
```

Рисунок 2.5. – Код у квантовому біт кодуванні

```
Chrombin = zeros(P, N);
for i = 1:P
    for j = 1:N
        pick = rand;
        if
pick > (QGAChrom(i, j)^2)
            Chrombit(i, j) = 1;
        else
            Chrombin(i, j) = 0;
        end
    end
end
```

Рисунок 2.6. – Перетворення квантового бітового коду в двійковий код

Рисунок 2.7. – Схема побудови одиничної хромосоми

2.4. Порівняння багатокритеріальних генетичних алгоритмів

В попередньому підрозділі ми описали певні багатокритеріальні генетичні алгоритми: VEGA, FFGA, NPGA, SPEA, SPEA-2, QGA. В цьому підрозділі ми порівняємо їх за перевагами, недоліками та їхніми особливостями (операторами). Для більш детальної інформації дивіться попередній підрозділ. В таблиці 2.2. порівняємо

Таблиця 2.2. Порівняння алгоритмів

Назва алгоритму	Переваги	Недоліки	Особливості
VEGA	Простота	Нездатність виробляти Парето-оптимальні рішення на опуклих поверхнях	Оператор селекції
FFGA	Простота	Налаштування радіусу ніші $-\sigma_{share}$	Ранг базоване визначення чи метод формування ніш для визначення пристосованості
NPGA	Простота з турнірною селекцією в розумінні, швидкість	Налаштування радіусу ніші, додатковий параметр для турнірної селекції дає складнішу реалізацію	Турнірна селекція, немає визначення пристосованості. Радіус ніші – механізм різноманітності
NSGA	Швидка збіжність	Налаштування радіусу ніші $-\sigma_{share}$	Пристосованість базується на сортуванні недомінантних рішень
SPEA	Добре тестований, кластеризація	Складний алгоритм кластеризації для підтримання різноманітності. Не гарантування збереження гран. рішень	Пристосованість базується на сортуванні недомінантних рішень в зовнішньому наборі. Кластеризація для відсікання зовнішньої популяції.
SPEA-2	Збереження граничних рішень	Складність в обчисленні пристосованості	Пристосованість базується на силі Парето (домінантних рішень). Щільність базується на методі k-найближчих сусідів. Наявний елітизм та зовнішній набір

Продовження таблиці 2.2.

QGA	мала кількість ітерацій, продуктивність	Копіювання суперпозиції, налаштування параметрів	Еволюція не полягає в оновленні операторів кросинговера та мутації. Нова концепція

Оператор селекції у VEGA алгоритмі є головною особливістю: у кожному поколінні генеруються ряд підгруп (субпопуляцій), які здійснюють пропорційну селекцію відповідно до кожного критерію. Таким чином, для проблеми з критеріями k генеруються підгрупи k розміром N/k (припускаючи загальний розмір популяції N). Ці підгрупи буде перемішано, щоб отримати нову популяцію розміром N, на яку застосовуватиметься ГА, в якому оператори кросинговеру та мутації використовуються звичайним способом.

Висновки до розділу

Поставлена задача структурно-параметричного синтезу нейронних мір із використанням генетичних алгоритмів. Описано генетичні алгоритми, їх основні властивості та оператори. Розглянуто основні генетичні алгоритми (VEGA, FFGA, NPGA, NSGA, SPEA, SPEA2, QGA) та відносно нові, але ефективні, їх переваги та недоліки.

РОЗДІЛ 3. БАГАТОКРИТЕРІАЛЬНА СИСТЕМА ОПТИМІЗАЦІЇ НАЛАШТУВАННЯ НЕЙРОННИХ МЕРЕЖ

3.1. Обгрунтування необхідності використання

Обчислювальні пристрої часто обмежені апаратними ресурсами з точки зору їх енергоспоживання, доступної пам'яті, доступних FLOP-операцій (англ. *Floating Point Operations Per Second* – операцій з рухомою комою за секунду») та обмежень часу. Отже, реальний дизайн моделі ШНМ необхідний для збалансування цих численних критеріїв (наприклад, точність та складність моделі). Часто, коли одночасно розглядаються кілька критеріїв проектування, може не існувати єдиного рішення, яке оптимально відповідає всім бажаним критеріям, особливо з конкуруючими критеріями. За таких обставин набір рішень, що забезпечує репрезентативну інформацію про компроміс між критеріями, є більш бажаним. Це дозволяє досліднику (користувачу) проаналізувати важливість кожного критерію, залежно від застосування, та вибрати відповідне рішення на кордоні компромісу для реалізації. Отже, пропонується багатокритеріальна система оптимізації налаштування нейронних мереж на основі багатокритеріальних генетичних алгоритмів, що були подані в розділі 2.

Актуальність даної системи і є обгрунтування створення. Ця система надає змогу оптимізувати процес розв’язку проблем високої складності, тобто виборі та визначенні оптимальних параметрів для нейронних мереж.

Застосування цієї системи на практиці є актуальним, оскільки при виборі оптимальних параметрів нейронної мережі ми розв’язуємо неймовірну кількість задач: починаючи із задач максимізації значення функції, розподілу оптимального часу до навчання роботів щодо польоту, ходьби і так далі. Безперечно, кожна задача має свій клас. Але для розв’язання цих задач нам необхідно оптимізувати параметри нейромережевих моделей, які ми вже згадували раніше. Це параметри як чисельність нейронів, чисельність шарів, вагові коефіцієнти для заданого типу задач. Сфера застосування штучних нейронних мереж є безмежною, а ця система дає спосіб оптимізувати цей процес.

Оскільки це багатокритеріальне завдання, яке представлене через певні часткові критерії f_k(x), k> 1 та їх набір $F(X)$ будемо називати векторним критерієм оптимальності, який належить до діапазону допустимих значень.

Оскільки це багатокритеріальне завдання, то існують деякі часткові рішення-критерії $f_k(x)$ при $k > 1$. У нашому випадку цими критеріями будуть параметри, які ми хочемо отримати на виході, а саме: типи алгоритму та його параметри (чисельність популяції, чисельність поколінь, тип оператора селекції, ймовірність виконання оператора мутації, ймовірність виконання оператора кросинговеру)). Звичайно, оскільки єдиного рішення для всіх часткових критеріїв немає, ми будемо шукати набір оптимальних значень Парето, виходячи з того, що рішення повинно бути не гіршим за всіма частковими критеріями і принаймні кращим за одним, як описано у попередньому розділі.

Наша основна задача – знаходження оптимальних рішень по Парето. Що дають можливість підібрати оптимальні алгоритми та їх оптимальні параметри. Для виконання цього ми маємо множину багатокритеріальних алгоритмів $A_1 \ldots A_n$ (n – кількість алгоритмів), та клас задач $T_1 \ldots T_q$ (q відповідає за певний тип задач для розв'язання), до кожної задачі є набір критеріїв задачі kr_1…k$_o$ та множина топологій нейронних мереж. Необхідно за вхідними навчальними та тестовими вибірками сформулювати нейромежережеву модель, в якої параметри будуть оптимально визначені за рахунок використання багатокритеріальних генетичних алгоритмів. Потрібно оптимальним чином обрати багатокритеріальний генетичний алгоритим, щоб мінімізувати наступні критерії структурно-параметричнного синтезу: точність та складність моделі (для зменшення ресурсів). Точність буде обчислюватися як значення середньої квадратичної квадратичної похибки на тестувальному наборі, а складність – кількість нейронів.

3.2. Постановка задачі класифікації

Як задачу для вирішення проблеми

Скажімо, у вас є магазин, і ви хочете знати, чи хоче хтось із ваших клієнтів знову прийти до вашого магазину чи ні. Відповідь на це запитання може бути лише «так» або «ні». Отакі проблеми в машинному навчанні розглядаються як класифікаційні проблеми. Проблеми класифікації, як правило, мають категоричний

результат, наприклад "так" або "ні", "1" або "0", "істинно" або "хибно". Для іншого прикладу, скажімо, ви хочете перевірити, чи можете ви грати в гольф у певний день чи ні. У цьому випадку погодні умови є важливими факторами, і на їх основі результат може бути або «Грати», або «Не грати».

Усі ці методи аналізу та моделювання даних можна розділити на дві групи: "навчання з викладачем" та "без викладача" . Але контрольоване навчання вимагає введення, яке має як незалежні змінні, так і залежну (залежну) змінну, значення якої потрібно оцінити. Використовуючи різні методи, процес «дізнається», як змоделювати (передбачити) значення цільової змінної на основі провісникових променів. Різні способи допомагають процесу "навчання", як передбачити значення цільової змінної на основі провісників променів.

Навчання без учителя не визначає залежну (цільову) змінну, але ставиться до всіх змінних однаково. У цьому випадку головною метою є не прогнозування значення змінної, а швидше пошук шаблонів, групувань чи інших способів характеристики даних, які можуть приводити до розуміння того, як дані взаємопов’язані. До прикладів навчання без наглядача ми можемо віднести кластерний аналіз, кореляціюфакторний аналіз, статистичні методи.

Класифікація – це техніка, яка є частиною так званого «навчання з викладачем». Оскільки навчання з викладачем означає передбачення вхідних значень на основі мітки або залежної змінної прикладів навчання, які ви надали раніше, ми можемо формально сформулювати проблему класифікації:

Нехай X – множина, Y– кінцева множина міток (номерів) класу. Існує певна невідома залежність цілей – відображення $y^*: X \to Y$, та значення відомі лише у підсумковій навчальній вибірці $X^m = \{(x_1, y_1), \dots, (x_m, y_m)\}$. Необхідно отримати алгоритм, здатний класифікувати будь-який об'єкт $x \in X$ [1].

За типами класів проблему класифікації поділяють на:

– двокласова класифікація;
– багатокласова класифікація;
– непересічні класи (класи, що не перетинаються);
– пересічні класи (класи, що перетинаються);
– нечіткі класи.

Двокласова класифікація – найпростіша класифікаційна задача, в якій кількість класів становить два.

Багатокласова класифікація – одна з найпростіших задач класифікації, коли кількість класів перевищує 2, наприклад, при розпізнаванні мови.

Класи, що перетинаються, вказують на те, що об’єкт може належати до кількох класів. Ті, що не перетинаються – об’єкт належить лише одному з класів.

Нечіткі класи – потрібно визначити ступінь належності об’єкта до кожного з класів, як правило, дійсне число від 0 до 1.

Як приклад багатокласової класифікації, на рис. 3.1. показаний приклад розпізнавання об’єктів на дорогах.

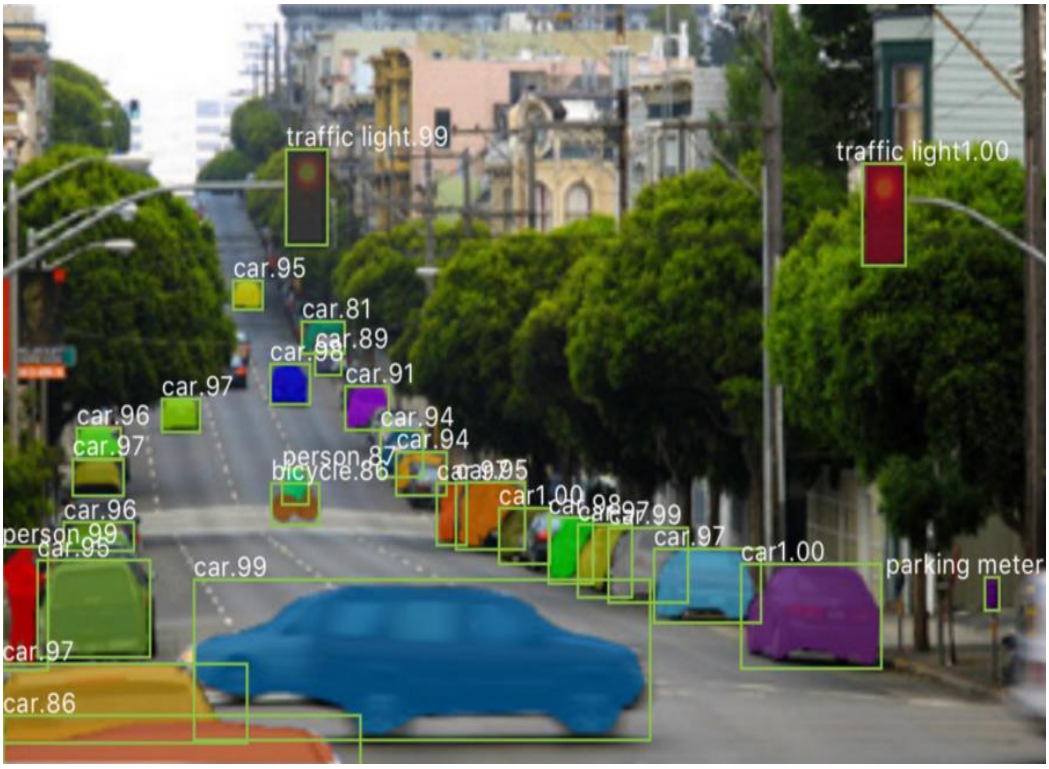


Рисунок 3.1. – Приклад класифікації дорожних об’єктів

3.3. Структура системи

Наступні компоненти системи оптимізації налаштування нейронних мереж для вирішення задачі класифікації складають нашу систему:

- компонент формування хромосоми;
- компонент багатокритеріальних генетичних алгоритмів;
- компонент навчання нейронних мереж.

На рис. 3.2. показано скорочену схему багатокритеріальної системи оптимізації налаштування нейронних мереж. Звичайно, під генетичним алгоритмами $A_1,\ldots,A_n$

маються на увазі багатокритеріальні генетичні алгоритми в попередніх розділах – VEGA, FFGA, NPGA, NSGA, SPEA, SPEA-2, QGA.

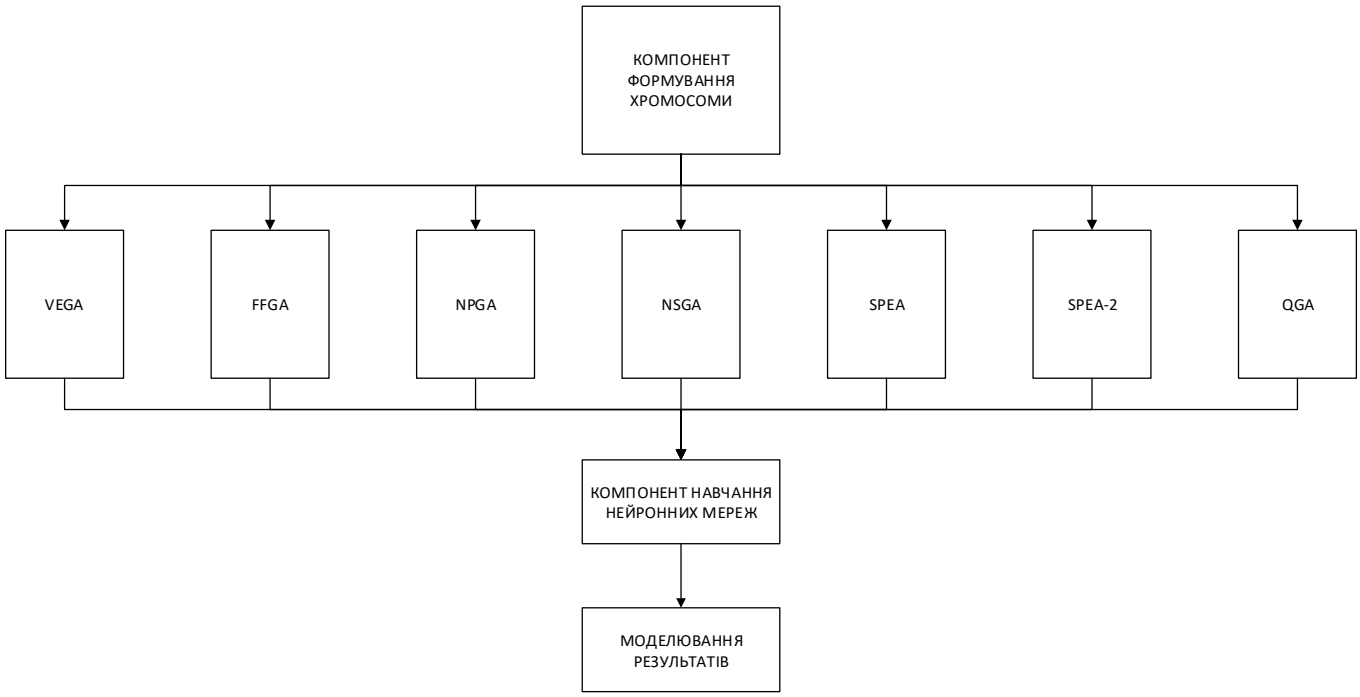


Рисунок 3.2. – Скорочена структурна схема багатокритеріальної системи оптимізації налаштування нейронних мереж

Повну структурну схему можна подивитися у додатку Б.

3.3.1. Компонент формування хромосоми

Першим етапом нашої системи буде формування хромосоми як частини популяції та тією частини системи де буде зберігатися інформацію про параметри штучних нейронних мереж. Цими параметрами є кількість шарів, кількість вхідних нейронів на відповідних шарах, вагові коефіцієнти

Звичайно, на хромосому накладаються певні обмеження. Хромосома буде кодуватися двома способами. Гени в звичайно випадку подається через бінарні числа («0» та «1»). На нашу хромосому ми накладаємо обмеження. Під кількість шарів ми виділимо 3 гени (тобто 3 розрядів). З цього виходить обмеження на кількість шарів – максимум 7 («111» у двійковій системі). Для кількості нейронів ми виділимо 10 х 7 генів = 70 генів. Максимум на кожному шарі = 1023. Вагові коефіцієнти будуть подаватися у іншому кодуванні – репрезентація чисел з плаваючою крапкою.

Наша хромосома буде мати наступні набори генів як на рис. 3.3. (кількість розрядів звичайно не показано):

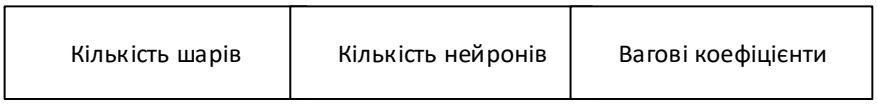


Рисунок 3.3. – Гени хромосоми

3.3.2. Компонент багатокритеріальних генетичних алгоритмів

Це найголовніший компонент нашої системи, що складається з паралельно з'єднаних багатокритеріальних алгоритмів, які виконують їхні базові функції – еволюцію хромосоми для досягнення рішення. На вхід відбувається подача уже сформованих хромосом.

Компонент багатокритеріальних генетичних алгоритмів – це 7 паралельно з'єднаних багатокритеріальних генетичних алгоритмів для отримання найкращих хромосом, що потім будуть декодовані та навчатися у компоненті навчання нейронних мереж.

Алгоритми детально показано у попередньому розділі. З їхніми алгоритмами

3.3.3. Компонент навчання нейронних мереж

Це компонент системи, який виконує етапи навчання нейронної мережі. Схему компоненту можна побачити на рис. 3.4.

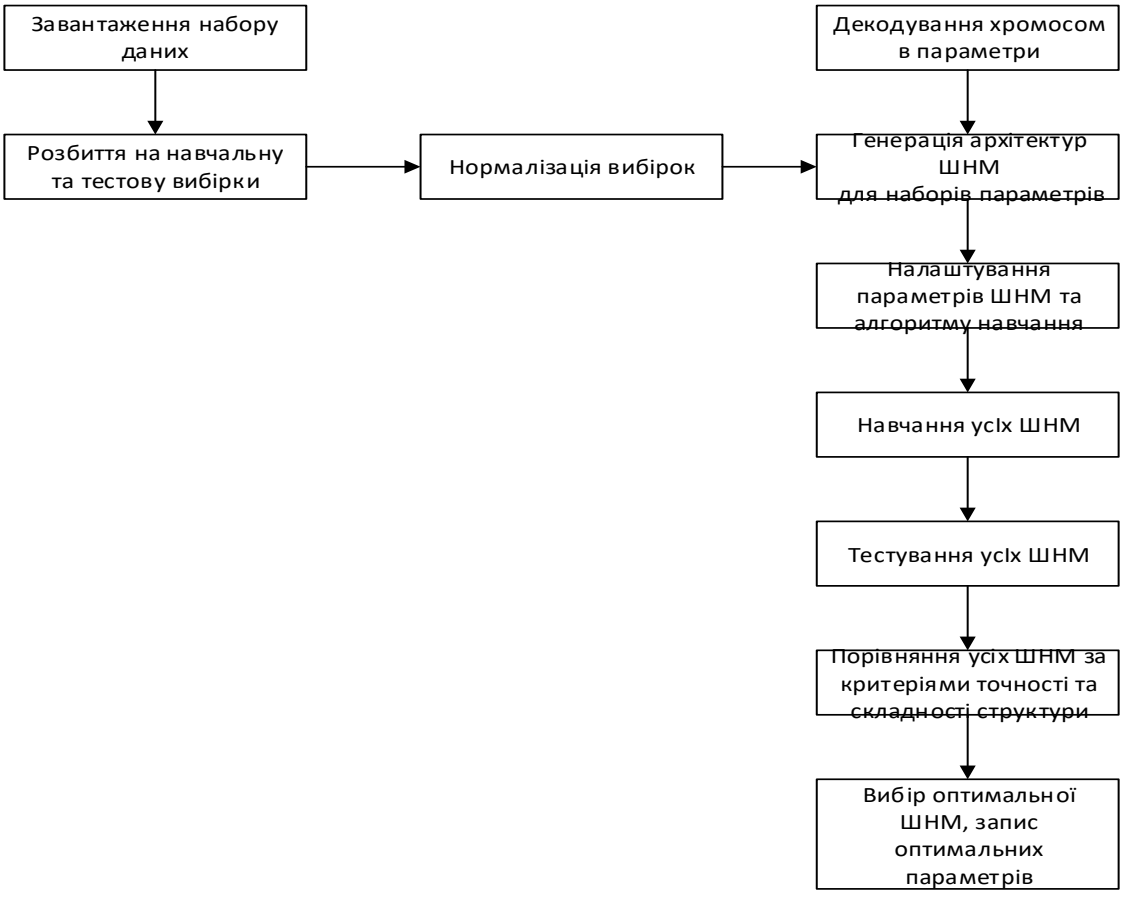


Рисунок 3.4. – Схема компоненту навчання нейронних мереж

Отже входом є усі хромосоми від 7 багатокритеріальних генетичних алгоритмів, які декодуються в параметри штучної нейронної мережі та задають структуру ШНМ. Далі проходяться основні етапи навчання нейронної мережі для усіх наборів, що показані на рис. 3.4., порівняння нейронних мереж за критеріями точності та складності. Точність в нашому випадку є точністю класифікації на тестовій вибірці. Складність структури – значення складності моделі, яке порівнюється по кількості шарів та нейронів в моделях.

Необхідно зазначити про використанні новітніх методів градієнтного спуска для навчання. В дисертації буде використовуватися найпоширеніший з новітніх методів градієнтного спуску для навчання нейронних мереж – adam.

Представляючи алгоритм, автори перелічують привабливі переваги використання adam в неопуклих задачах оптимізації [57], наступним чином:

- пряма реалізація;
- обчислювальна ефективність;
- невеликі вимоги до пам’яті;
- інваріант діагональному масштабуванню градієнтів;
- добре підходить для великих проблем з точки зору даних та / або параметрів;
- підходить для нестаціонарних цілей;
- підходить для проблем із дуже галасливими або рідкісними градієнтами;
- гіперпараметри мають інтуїтивну інтерпретацію і, як правило, вимагають незначної настройки.

Adam різниться з стандартним стохастичним алгоритмом.

Стохастичний градієнтний спуск підтримує єдину швидкість навчання (називається $alpha$) для всіх оновлень ваги, і швидкість навчання не змінюється під час тренування.

Швидкість навчання підтримується для кожної ваги мережі (параметра) і окремо адаптується в міру розгортання навчання. Метод обчислює індивідуальні показники адаптивного навчання для різних параметрів на основі оцінок першого та другого моментів градієнтів. Автори описують adam як поєднання переваг двох інших розширень стохастичного градієнтного спуску. Зокрема:

- адаптивний градієнтний алгоритм (AdaGrad) [57], який підтримує швидкість навчання за параметром, що покращує ефективність при проблемах з розрідженими градієнтами (наприклад, проблеми з природною мовою та комп'ютерним зором);
- розмноження середньоквадратичного корінця (RMSProp) [57], яке також підтримує коефіцієнти навчання за параметрами, які адаптуються на основі середнього значення останніх величин градієнтів для ваги (наприклад, наскільки швидко вона змінюється). Це означає, що алгоритм добре справляється з онлайновими та нестаціонарними проблемами (наприклад, шумними).

Adam усвідомлює переваги як AdaGrad, так і RMSProp. Замість того, щоб адаптувати швидкість навчання параметрів на основі середнього першого моменту (середнього значення), як у RMSProp, Адам також використовує середнє значення других моментів градієнтів (нецентрова дисперсія).

Зокрема, алгоритм обчислює експоненціальну ковзну середню градієнта та квадратичний градієнт, а параметри beta1 та beta2 контролюють швидкість спаду цих ковзних середніх.

Початкове значення ковзних середніх та значень $beta1$ та $beta2$, близьких до 1.0 (рекомендовано), призводять до зміщення оцінок моменту до нуля. Це упередження долається спочатку обчисленням упереджених оцінок, а потім обчисленням, скоригованим із зміщенням.

Параметри adam:

- $alpha$. Також називається швидкістю навчання або розміром кроку. Відсоток оновлення ваг (наприклад, 0,001). Більші значення (наприклад, 0,3) призводять до швидшого початкового навчання перед оновленням курсу. Менші значення (наприклад, 1.0E-5) сповільнюють навчання прямо під час тренування;
- $beta1$. Експоненціальна швидкість занепаду для першого моменту оцінюється (наприклад, 0,9);

- $beta2$. Експоненціальна швидкість занепаду для оцінок другого моменту (наприклад, 0,999). Це значення слід встановити близько 1,0 для проблем з розрідженим градієнтом (наприклад, проблеми комп'ютерного зору);
- epsilon. Це дуже мала кількість для запобігання будь-якому діленню на нуль у реалізації (наприклад, 10E-8).

Хорошими налаштуваннями за замовчуванням для перевірених проблем машинного навчання є alpha = 0,001, beta1 = 0,9, beta2 = 0,999 та epsilon = 10−8 [57].

3.4. Алгоритм роботи системи

Увесь цей процес генетичних алгоритмів, що бере участь в підсистемі обчислення залежить від включених в неї алгоритмів. Ці всі методи мають свою певну систему визначення оптимальних параметрів – тобто значень генів нашої хромосоми).

Як наслідок, необхідно визначити конкретно оптимальний або оптимальні ГА під вибраний тип задачі. Алгоритм починається з заповнення хромосоми, яку структуру ми вже сформували. Нехай R_k – кількість розрядів під кількість шарів ШНМ, R_m – кількість розрядів під кількість нейронів кожного шару, R_w – кількість розрядів, що відповідають за відповідні вагові коефіцієнти нейронів з шарів.

Кроки алгоритму:

1. Ініціалізація хромосоми, що показана на рис. 3.3. та виділення розрядів під ті гени.
2. Обчислення чисельності популяції та кількості поколінь ГА (виходячи з того, що ці параметри залежать від розміру хромосоми).
3. Ініціалізація – формування хромосом, кількість яких дорівнює розміру популяції та заповнення генів хромосом випадковими бітами за допомогою генератору випадкових чисел.
4. Копіювання популяції в N багатокритеріальних генетичних алгоритмів.
5. Для всіх генетичних алгоритмів: Розрахунок пристосованості хромосом. Якщо еволюція не завершена, то необхідно перейти до наступного кроку. Інакше – крок 8.

6. Усі генетичні алгоритми мають наступне: еволюція за допомогою операторів ГА (згідно конкретного алгоритму). Тобто оператори селекції, кросинговеру, мутації.
7. Усі генетичні алгоритми мають наступне: одержання нового покоління розміром в популяцію.
8. Кінець еволюції та одержання оптимальних параметрів (кількість шарів, кількість нейронів для усіх шарів, відповідні вагові коефіцієнти) в виді хромосом з усіх алгоритмів, які йдуть на навчання моделей серед яких вибирається найкраща з моделей. Процес навчання показано в розділі 3.3.3. я компонент навчання нейронних мереж.

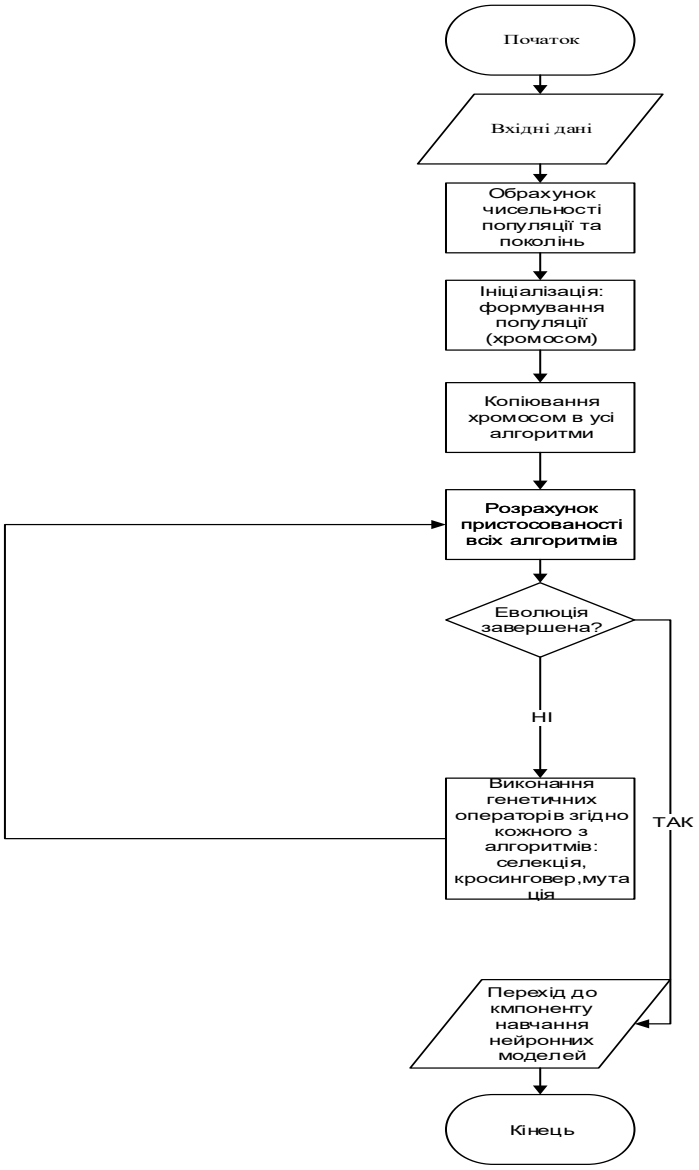


Рисунок 3.5. – Алгоритм роботи системи

3.5. Створення бази даних

Можливі БД для застосування: SQLite, MongoDB, SQLite, PostgreSQL.

Якщо говорити про MySQL, то це перевірена часом та стійка реляційна база даних, що зручна для зберігання великої кількості даних, що пов’язані між собою певними типами зв’язків. PostgreSQL, SQLite відносяться до того ж самого типу задач. Проте перша має гігантську кількість додаткових типів даних, можливостей та налаштувань. SQLite – полегшена та мобільна версія MySQL. Останній же варіант є прикладом так званих NoSQL-баз даних, які засновані на методі зберігання даних {ключ: значення}. Вони є зручними для зберігання великої кількості даних, що між собою пов'язані можуть бути слабко, однак виграє у швидкості на голову.

У нашій роботі буде використовуватися реляційну БД з одною таблицею – для збереження інформації про стан параметрів нейронної мережі для певного багатокритеріального генетичного алгоритму, тобто кількість шарів, кількість нейронів, значення вагових коефіцієнтів. Ця таблиця, що й буде собою являти базу даних показана в табл.3.1.

Таблиця 3.1. – Структура таблиці багатокритеріальних генетичних алгоритмів.

#	Опис	Назва	Тип даних	Примітка	Обозв’язковість заповнення
1	Ідентифікатор багатокритеріального ГА	ID_MGA_	INT	–	Так
2	Ім’я алгоритму	NAME_MGA_	VARCHAR (256)	–	Так
3	Чисельність шарів	NUM_LAYER	INT	–	Так
4	Чисельність нейронів	NUM_NEURON	INT	–	Так
5	Вагові коефіцієнти	NUM_WEIGHT	VARCHAR (65535)	Зберігаються у вигляді строки	Так

Висновки до розділу

Обґрунтовано необхідність багатокритеріальної системи оптимізації налаштування нейронних мереж. Необхідність пошуку більш ніж одного критерію пов’язана з тим, що всяка практична задача вимагає певного вибору та відповідного прийняття рішень. Але в процесі ми стикуємося з багатьма альтернативами або

критеріями якості. В процесі часто не можливо адекватно виразити усі ці критеріїв якості через один, який буде згорткою усіх. Отже для вирішення таких задач розглядаються альтернативи і вибираються оптимальні рішення. Це рішення може бути одним, але часто – не одне.

Подано задачу класифікації як спосіб перевірки багатокритеріальної системи оптимізації налаштування нейронних мереж.

Побудовано структури багатокритеріальної системи оптимізації налаштування нейронних мереж та запропоновано алгоритм роботи системи. Дана система складається з 3 основних компонентів (блоків), а саме – компоненту формування хромосоми, компоненту багатокритеріальних алгоритмів, компоненту навчання нейронних мереж.

РОЗДІЛ 4. РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Функціонал ПЗ

Відповідно до розділу 3 ми маємо наступні компоненти (блоки):

- компонент формування хромосоми;
- компонент багатокритеріальних генетичних алгоритмів;
- компонент навчання нейронних мереж.

В розділі 3 оговорено основні блоки багатокритеріальної системи оптимізації налаштувань нейронних мереж. Схема системи розміщена у додатку Б. Виходячи з того, програма повинна виконувати наступні функції:

- зчитування датасету;
- форматування датасету до необхідного формату;
- вхідні дані повинні розбиватися на відповідні тренувальні та тестові набори;
- виконувати певні багатокритеріальні генетичні алгоритми для знаходження оптимальних параметрів нейромережевих моделей;
- навчання моделей;
- порівняння результатів та експорт параметрів моделі.

При цьому в додатку В побудована блок-схема основних етапів будь-якого генетичного алгоритму

4.2. Опис вхідних та вихідних даних та інтерфейс

Вхідні та вихідні дані є в кожному компоненті.

Першим етапом нашої системи буде формування хромосоми як частини популяції та тією частини системи де буде зберігатися інформацію про параметри штучних нейронних мереж. Цими параметрами є кількість шарів, кількість вхідних нейронів на відповідних шарах, вагові коефіцієнти. Це зберігається на в компоненті формування хромосоми. Обмеження системи було описано в попередньому розділі. На виході компонету ми й отримуємо хромосому, що буде передаватися як вхід на компоненту багатокритеріальних генетичних алгоритмів для подальшої її через алгоритми, які описано в розділі 2. На виході компоненту багатокритеріальних

генетичних алгоритмів ми отримуємо найкращі хромосоми, що передаються в компонент навчання нейронних мереж.

В компонент навчання нейронних мереж входом є усі хромосоми від 7 багатокритеріальних генетичних алгоритмів, які декодуються в параметри штучної нейронної мережі та задають структуру ШНМ.

Для вводу даних навчальної та тестової вибірки ми використовуємо компонент навчання нейронних мереж. Також в цьому компоненті ми вибираємо класи задачі (але так як розв'язується задача класифікації, то лише одна), команди програми, що складають наш інтерфейс. Основними класами задач ми вважаємо класифікацію, апроксимацію, задачу прийняття рішень, задачу прогнозування, управління, а також додаткове включення чи виключення багатокритеріальних алгоритмів. Але в цій задачі ми розглядаємо задачу класифікацію тільки.

Основним виходом компоненту навчання нейронних мереж є параметри нейронної мережі, результати кожного з багатокритеріальни генетичних алгоритмів.

Після основних алгоритмічних компонентів ми робимо порівняння алгоритмів за точністю та складністю та моделюємо результати.

Рисунок 4.1. – Скорочений вигляд програмної реалізації

В додатку Б систему можна переглянути більш детально

Якщо розглядати інтерфейс, то він реалізується за допомогою командної строки. У таблиці 4.1. подано усі аргументи для програми:

Таблиця 4.1. Аргументи з їх можливими значеннями та обов'язкістю

Ключ	Можливі значення	Обов'язково
action	{build, build_n_train}	Обов'язково
target	{0,1}	Обов'язково
dataset	Шлях до директорії з датасетом	Обов'язково
search	{0, 1}	Необов'язково
save_to	Шлях до директорії для збереження результатів	Необов'язково
vega	{0, 1}	Необов'язково
ffga	{0, 1}	Необов'язково
npga	{0, 1}	Необов'язково
nsga	{0, 1}	Необов'язково
spea	{0, 1}	Необов'язково
spea2	{0, 1}	Необов'язково
qga	{0, 1}	Необов'язково
ts-r2ea	{0, 1}	Необов'язково
epochs	ціле число	Необов'язково
batch_size	ціле число	Необов'язково
lr	крок навчання	Необов'язково
opt	{sgdm, nag, rmsprop, adagradient, adam}	Необов'язково
help	{0, 1}	Необов'язково

Параметр «save_to» застосовується у випадку коли користувач має на меті зробити збереження результатів роботи системи у директорію, що відрізняється від директорії за замовчуванням.

Аргументи «hiddens_», «epochs_», «batch_size_», «lr_», «opt_» є необов'язковими та необхідні для зміни значень за замовчуванням параметрів функціонування системи. Аргументи «vega_», «ffga_», «npga_», «nsga_», «spea_», «spea2_», «qga_», «ts-r2ea_» відповідаються за відключення або виключення багатокритеріальних генетичних алгоритмів. Приклад багатошарової моделі з 1 прихованим шаром можна побачити у додатку Г.

4.3. Контрольний приклад на MNIST-датасеті

Це базовий алгоритм для перевірки нейронних мережі[50]. Складається з 60000 тренувальних даних та 10000 тестових даних.

Кожне зображення має висоту 28 пікселів і ширину 28 пікселів, загалом 784 пікселі. З кожним пікселем пов'язане одне значення пікселя, що вказує на світлість або темність цього пікселя, а більші цифри означають темніше. Це піксельне значення – ціле число від 0 до 255 включно.

Набір навчальних даних (train.csv) має 785 стовпців. Перший стовпець, який називається "мітка", – це цифра, яку намалював користувач. Решта стовпців містять піксельні значення відповідного зображення. На рис. 4.2. показано зразок MNIST датасету.

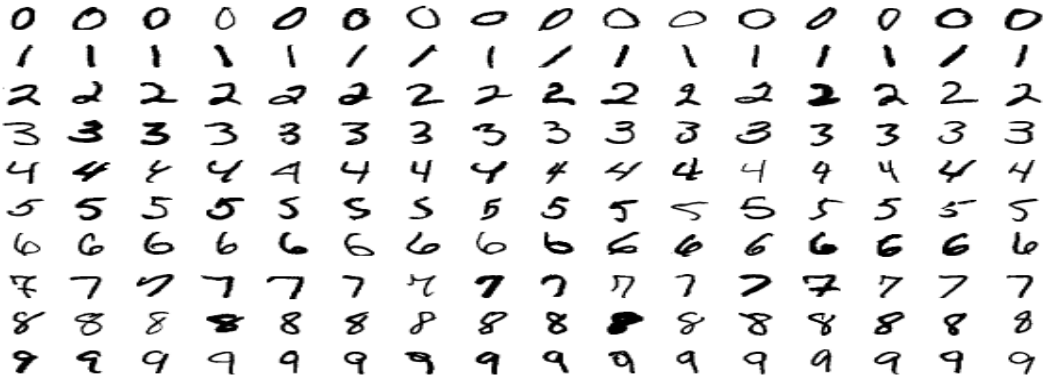


Рисунок 4.2. – Зразок MNIST датасету

Для отримання результатів використовувалися всі багатокритеріальні алгоритми, які включені в програму, тобто VEGA, FFGA, NPGA, NSGA, SPEA, SPEA2, QGA, TS-R2EA. В таблиці 4.2. показано результати точності при заданих критеріях точності та складності моделі.

Таблиця 4.2. Результати MNIST датасету

Алгоритм	Точність
VEGA (зеленуватий)	98.53%
FFGA (orange)	99.17%
NPGA (light blue)	99.18%
NSGA (бордовий)	99.07%
SPEA (pink)	99.05%
SPEA2 (red)	99.45%
QGA (сірий)	98.88%

При цьому оптимальна по складності та точності виявилася відносно простий багатошаровий перцептрон з 3 прихованими шарами, що показано на рис. 4.3.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_27 (Dense)             (None, 364)               285740
_________________________________________________________________
batch_normalization_13 (Batc (None, 364)               1456
_________________________________________________________________
dropout_13 (Dropout)         (None, 364)               0
_________________________________________________________________
dense_28 (Dense)             (None, 52)                18980
_________________________________________________________________
batch_normalization_14 (Batc (None, 52)                208
_________________________________________________________________
dropout_14 (Dropout)         (None, 52)                0
_________________________________________________________________
dense_29 (Dense)             (None, 10)                530
=================================================================
Total params: 306,914
Trainable params: 306,082
Non-trainable params: 832
```

Рисунок 4.3. – Параметри оптимальної моделі

У додатку Д можна розглянути найкращі результати різних нейронних мереж на даному MNIST датасеті серед турніру на Kagle. Реальне найбільше значення точності наближається до 99.70%, але та структура є дуже складною та потребує великих обчислювальних потужностей. У нас же постає задача про оптимізацію. Не кажучи вже про те, що є точність моделі не є головним показником. Оскільки деякі моделі просто не видають хороших результатів на інших датасетах. А в додатку Е показано графічний вид результатів.

Висновки до розділу

Відповідно до структури з розділу 3 розроблено ПЗ з необхідним функціоналом, який детально описано в цьому розділі. Описано вхідні та вихідні дані, команди інтерфейсу.

В результаті програма була протестована на MNIST датасеті. В результаті оптимізаційна модель з SPEA-2 виявила найкращі результаті – 99,45%, що є кращим на 0.05% ніж за найкраще існуюче рішення (пакет TPOT[55]), що використовує генетичні алгоритми для налаштування нейронних мереж.

РОЗДІЛ 5. МАРКЕТИНГОВИЙ АНАЛІЗ СТАРТАП-ПРОЕКТУ

5.1. Опис ідеї проекту

Таблиця 5.1. Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення багатокритеріальної системи для оптимізації параметрів нейронних мереж при вирішенні проблем високої складності: задач класифікації, апроксимації, прогнозування, прийняття рішень, управління, які зустрічаються	Системи, що включать нейронні мережі: системи розпізнавання обличчя, інформаційна система пожежного спостереження та ін.	Поліпшення якості, швидкості та зручності настроювання параметрів ШНМ.

Таблиця 5.2. Визначення характеристик ідеї проекту

№ п/п	Техніко-економічні характеристики ідеї	Потенційні продукти конкурентів		W (слабка сторона)	N (нейтр. сторона)	S (сильна сторона)
		Мій	ТРОТ			
1	Кількість видів задач до вирішення	5	2			+
2	Кількість алгоритмів	Десяток	1			+
3	Зручність	Зручний інтерфейс	Не зрозумілий інтуїтивно		-	
4	Точність	Висока	Висока			+

5.2. Технологічний аудит ідеї проекту

Таблиця 5.3. Технологічна здійсненність ідеї проекту

№ п/п	Складові ідеї проекту	Технології її реалізації	Наявність технологій	Доступність технології
1	Обробка даних	Python, PyTorch, Numpy	+	Загально-доступні
2	Організація пошуку оптимальних параметрів моделей у просторі варіантів	Python, PyTorch, Numpy	+	Загально-доступні
3	Бібліотеки алгоритмів	PYGMO, Platypus	+	Загально-доступні

Продовження таблиці 5.3.

4	Моделювання та аналіз результатів	Python, PyTorch, Numpy, Matplotlib	+	Загально-доступні

У результаті технологічного аналізу ідеї проекту було вирішено для її реалізації використати наступний технологічний стек:

– мова програмування та віртуальна машина Python;
– фреймворк глибокого навчання PyTorch;
– бібліотека тензорних обчислень NumPy;
– бібліотека візуалізації Matplotlib;
– фреймворк досліджень Jupyter.

5.3. Аналіз ринкових можливостей запуску стартап-проекту

Таблиця 5.4. Попередня характеристика потенційного ринку стартап-проекту

№ п/п	Показники стану ринку	Характеристика
1	Кількість конкурентів	5
2	Загальний обсяг продаж	2 млн. ум. од.
3	Динаміка ринку (якісна оцінка)	Зростає
4	Наявність обмежень для входу та їх характер	Немає
5	Специфічні вимоги до стандартизації та сертифікації	Немає
6	Середня норма рентабельності по ринку	20%

Таблиця 5.5. Характеристика потенційних клієнтів стартап-проекту

Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
Необхідність зменшення загрози пожеж	МНС, підприємства	Масштаби застосування, обсяги даних, технічні засоби для розгортання, фінанси	Унікальність системи, більші можливості як зменшений час реагування, розрахунок масштабу пожежі, можливість настроювання.

Таблиця 5.6. Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція
1	Нестача технічнихре сурсів	Підприємства можуть мати обмежені обчислювальні ресурси для розгортання програмного забезпечення	Винесення системи на хмарні технології
2	Відсутність тренувальних наборів даних для моделювання	Замовник може не мати необхідного набору даних	Додавання програмного компонента, що дозволяє напівавтоматично зібрати достатній обсяг даних та частково їх розмітити
3	Відсутність універсальних інтерфейсів для роботи із системою	Інформаційні системи можуть бути не готові до інтеграції із продуктом	Створення універсального інтерфейсу для усіх платформ
4	Криза	Зменшення продажу	Спрощення комплектації системи. Залишаються тільки необхідні датчики та прилади для основного контролю та моніторингу без додаткових звітів, можливостей оптимального шляху евакуації.

Таблиця 5.7. Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція
1	Хмарні обчислення	Виконання розрахунків на віддалених серверах	Перенесення системи у хмарні сервісні середовища (Azure)
2	Сторонні сервіси обробки даних	Сторонні сервіси надають послуги розмітки наборів даних на вимогу	Надання автоматичної взаємодіхх з сервісами розмітки датасетів
3	Спрощення комплектації	Спрощення комплектації для зменшення ціни та націлення системи тільки на певні критерії необхідні замовнику	Залишаються тільки необхідні датчики та прилади для основного контролю та моніторингу без додаткових звітів

Таблиця 5.8. Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції – вільна конкуренція	Присутня невелика кількість постачальників. Домінуючих конкурентів немає через галузеву специфіку. Невелика кількість конкурентів та домінуючих конкурентів немає	Збільшення функціоналу та тісна співпраця з замовниками(клієнтами). Покращення технологічної частини.
2. За рівнем конкурентної боротьби – глобальний	Продукт не залежить від локації, хоч і більш доступний для України	
3. За галузевою ознакою – внутрішньогалузева	Продукт спрямований на оптимізації виробничих процесів замовника	
4. Конкуренція за видами товарів: – за необхідністю	Надавання сервісів відбувається за необхідністю	
5. За характером конкурентних переваг – нецінова	Ефективність – переваги	
6. За інтенсивністю – не марочна	Переваги не залежать від торгової марки	

Таблиця 5.9. Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти
	BigML OptiML	TPOT
Висновки	Контролюють значну частину ринку, мають узагальнені рішення Значна частина ринку у цих компаній, узагальненість рішень	Розрахунок йде на більш освічену аудиторію, яка знайома з машинним навчанням

Таблиця 5.10. Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор	Обґрунтування
1	Універсальність розрахунку	Універсальна, та висока достовірність отриманих результатів, яка підтверджується якісними математичними розрахунками.
2	Простота реалізації	Можливість установки на старі пульти пожежного спостерження з невеликими затратами.
3	Час передачі даних	Швидко передає результати до підрозділів МНС чи інших структур
4	Додаткові можливості	Не тільки спостереження, а й пошук оптимального шляху евакуації – потенційно врятує більше людських життів.

Таблиця 5.11. Порівняльний аналіз сильних та слабких сторін RFS

№ п/п	Фактор	Бали 1-20	Рейтинг продуктів-конкурентів у порівнянні з RFS						
			-3	-2	-1	0	+1	+2	+3
1	Висока якість	18	-	-	-		1	1	2
2	Оптимальне співвідношення ціни і якості	16	-	-	-		1	1	1
3	Додаткові можливості	20	-	-	-		1	1	2

Таблиця 5.12. SWOT-аналіз стартап-проекту

Сильні сторони: якість, універсальність, точність	Слабкі сторони: необхідність інвесторів, дешеві замінники.
Можливості: удосконалення	Загрози: додаткові витрати, розвиток конкурентів

Таблиця 5.13. Альтернативи ринкового впровадження стартап-проекту

№ п/п	Альтернатива (орієнтовний комплекс заходів)	Ймовірність отримання ресурсів	Терміни реалізації
1	Спеціалізовані рішення	Висока	1-3 місяці
2	Хмарний сервіс	Висока	3-6 місяців

Продовження таблиці 5.13.

3	Узагальнення рішення, вихід на нові сфери ринку	Середня	6-12 місяців

5.4. Розробка ринкової стратегії проекту

Таблиця 5.14. Вибір цільових груп потенційних споживачів

№ п/п	Опис цільової групи потенц. клієнтів	Готовність споживача сприйняти продукт	Орієнтовний попит в межах цільової групи	Інтенсивність конкурентів в сегменті	Простота входу у сегмент
1	МНС України (цільова група)	Висока готовність сприйняти товар. Оскільки існує потреба в якісній інформаційній системі, що спростить роботу пожежно-рятувальних підрозділів	Попит високий через те, що рівень ціни-якості є кращим чим у конкурентів	Товари-замінники конкурують з нашим товаром, але впринципі складно замінити цей товар товарами-замінниками, тому інтенсивність є слабка	Просто заходить. Оскільки цей товар кращий за замінник та аналогів не має.
2	Підприємства	Висока готовність для збільшення безпеки підприємства та зменшення збитків у разі виникнення пожеж.	Попит високий через те, що рівень ціни-якості є кращим чим у конкурентів	Товари-замінники конкурують з нашим товаром, але впринципі складно замінити цей товар товарами-замінниками, тому інтенсивність є слабка	Просто заходить. Оскільки цей товар кращий за замінник та аналогів не має.

Таблиця 5.15. Визначення базової стратегії розвитку

Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкуренто-спроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
Залучення закордонних виробників	Робить ставку на унікальність системи, більші можливості.	Відмітні властивості товару	Стратегія диференціації

Таблиця 5.16. Визначення базової стратегії конкурентної поведінки

Чи є проект «першопрохід-цем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
Ні	Забирати 20%	Можливість системи передавати результати до МНС	Стратегія виклику лідера

Таблиця 5.17. Визначення стратегії позиціонування

Вимоги до продукту цільової аудиторії	Базова стратегія розвитку	Ключові конкурентні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту
Робить ставку на унікальність системи, більші можливості як зменшений час реагування, розрахунок масштабу пожежі, можливість настроювання	Стратегія диференціації	Відмітні властивості товару	Надійність, швидкість передачі інформації про пожежу, багатофункціональність.

5.5. Розроблення маркетингової програми стартап-проекту

Таблиця 5.18. Визначення ключових переваг концепції потенційного продукту.

№ п/п	Потреба	Вигода, яку пропонує продукт	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Унікальність	Система передає дані про пожежі до підрозділів МНС та створює оптимальний план евакуації людей	Аналогів не існує. Є дешеві замінники, які виконують роль передачі даних до підрозділів МНС. Такими є пульти пожежного спостереження.
2	Час реагування	Час реагування менший	Оскільки товари-замінники використовують телефонні лінії, які постійно зайняті, то інформаційна система передбачає використання передачі інформації через мережу, що значно пришвидшує обробку даних.
3	Розрахунок масштабу пожежі	Створення звіту по пожежі	Створення звіту по пожежі, масштабу пожежі, мінімальна кількість ресурсів, яка потрібно для гасіння.

Таблиця 5.19. Визначення меж встановлення ціни

Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
100 у.о.	Немає аналога	Бюджет 12,6 млрд грн.	90-150 у.о.

Таблиця 5.20. Формування системи збуту

Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник продукту	Глибина каналу збуту	Оптимальна система збуту
Орієнтація на регулярні поставки	Встановлення контактів із споживачами та підтримка їх. Формування попиту. Дослідницька в області маркетингу.	без посередників	Серед МНС, підприємств.

Таблиця 5.21. Концепція маркетингових комунікацій

Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуються цільові клієнти	Ключові позиції, обрані для позиціону-вання	Завдання рекламного повідомлення	Концепція рекламного звернення
Орієнтація на регулярні поставки	Формальні/неформаль ні канали комунікацій	Універсальність, час реагування на пожежі, розрахунок масштабу пожежі, доступність	Інформування споживачів; Стимулювання продажу;	Купуючи цю систему – ви отримуєте надійного пожежника!

Висновки до розділу

Виходячи з вищенаведених результатів можна говорити про наявність попиту на заявлену систему. Необхідно додати про присутність малої конкуренції, оскільки не існує товару-аналога для використання у вузькій галузі, що збільшує конкурентоспроможність проекту.

ВИСНОВКИ

Багатокритеріальна система оптимізації налаштування нейронних мереж оптимізує розв’язання задач з великою кількістю критеріїв. При цьому необхідність пошуку багатьох критеріїв є актуальною. Будь-яка практична задача вимагає вибору та прийняття рішення про вибір однієї або декількох альтернативник рішень серед переліку тих. І, як правило, дуже складно виразити усі ці критерії через один, що буде згорткою усіх. Тому й розглядається вибір (селекція) оптимальних альтернатив та обираються найкращі рішення.

Актуальність даної системи і є обгрунтування створення. Ця система надає змогу оптимізувати процес розв’язку проблем високої складності, тобто виборі та визначенні оптимальних параметрів для нейронних мереж, що можуть застосовуватися для розпізнавання зображень, польоту літаків, менеджмент часу, пожеж, тощо. В магістерській дисертації систему було протестовано на розв’язанні задачі класифікації.

Розроблено ПЗ, яке включає систему налаштування нейронних мереж, що базується на використанні 7 багатокритеріальних генетичних алгоритмів. Контрольний приклад показав, що точність класифікації в 99,45% (що на 0.05% більше, ніж найкраще існуюче рішення – TPOT) на вибраному наборі даних досягається з оптимально простою за складністю моделлю, адже модель допомагає дослідникам скоротити час на створення дієвої моделі, що видає високі практичні результати. Що й каже про потенціал цієї системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. C.A.C. Coello, Evolutionary multi-objective optimization: a historical view of the field, Comput. Intell. Mag. IEEE 1 (1) (2006) 28–36.
2. K. Deb, Multi-Objective Optimization Using Evolutionary Algorithms, vol. 16, John Wiley & Sons, 2001.
3. A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P.N. Suganthan, Q. Zhang, Multi-objective evolutionary algorithms: a survey of the state of the art, Swarm Evol. Comput. 1 (1) (2011) 32–49.
4. B. Li, J. Li, K. Tang, X. Yao, Many-objective evolutionary algorithms: a survey, ACM Comput. Surv. 48 (1) (2015) 1–35.
5. H. Ishibuchi, N. Tsukamoto, Y. Nojima, Evolutionary many-objective optimization: a short review, Proceedings of the IEEE Congress on Evolutionary Computation (2008) 2419–2426.
6. M. Farina, P. Amato, A fuzzy definition of “optimality” for many-criteria optimization problems, IEEE Trans. Syst. Man Cybern. Part A: Syst. Hum. 34 (3) (2004) 315–326.
7. M. Кцppen, R. Vicente-Garcia, B. Nickolay, Fuzzy-Pareto-dominance and its application in evolutionary multi-objective optimization, in: Proceedings of the Evolutionary Multi-criterion Optimization, Springer, 2005, pp. 399–412.
8. S. Yang, M. Li, X. Liu, J. Zheng, A grid-based evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 17 (5) (2013) 721–736.
9. R. Wang, R.C. Purshouse, P.J. Fleming, Preference-inspired coevolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 17 (4) (2013) 474–494.
10. M. Li, S. Yang, X. Liu, Shift-based density estimation for Pareto-based algorithm in many-objective optimization, IEEE Trans. Evol. Comput. 18 (3) (2014) 348–365.

11. X. Zhang, Y. Tian, Y. Jin, A knee point-driven evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 19 (6) (2015) 761–776.
12. T. Murata, M. Gen, Cellular genetic algorithm for multi-objective optimization, in: Proceedings of the 4th Asian Fuzzy System Symposium, Citeseer, 2002, pp. 538–542.
13. Q. Zhang, H. Li, MOEA/D: a multiobjective evolutionary algorithm based on decomposition, IEEE Trans. Evol. Comput. 11 (6) (2007) 712–731.
14. H.L. Liu, F. Gu, Q. Zhang, Decomposition of a multiobjective optimization problem into a number of simple multiobjective subproblems, IEEE Trans. Evol. Comput. 18 (3) (2014) 450–455.
15. K. Li, K. Deb, Q. Zhang, S. Kwong, An evolutionary many-objective optimization algorithm based on dominance and decomposition, IEEE Trans. Evol. Comput. 19 (5) (2015) 694–716.
16. M. Li, S. Yang, X. Liu, Pareto or non-Pareto: bi-criterion evolution in multiobjective optimization, IEEE Trans. Evol. Comput. 20 (5) (2016) 645–665.
17. Y. Liu, D. Gong, J. Sun, Y. Jin, A many-objective evolutionary algorithm using a one-by-one selection strategy, IEEE Trans. Cybern. (2018) (in press).
18. M. Fleischer, The measure of Pareto optima applications to multi-objective metaheuristics, in: Proceedings of the Evolutionary Multi-criterion Optimization, Springer, 2003, pp. 519–533.
19. E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271.
20. N. Beume, B. Naujoks, M. Emmerich, SMS-EMOA: multiobjective selection based on dominated hypervolume, Eur. J. Oper. Res. 181 (3) (2007) 1653–1669.

21. J. Bader, E. Zitzler, HypE: an algorithm for fast hypervolume-based many-objective optimization, Evol. Comput. 19 (1) (2011) 45–76.
22. H. Wang, L. Jiao, X. Yao, Two Arch2: an improved two-archive algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 19 (4) (2015) 524–541.
23. C.A. Rodrиguez Villalobos, C.A.C. Coello, A new multi-objective evolutionary algorithm based on a performance assessment indicator, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2012, pp. 505–512.
24. H. Ishibuchi, H. Masuda, Y. Tanigaki, Y. Nojima, Modified distance calculation in generational distance and inverted generational distance, EMO (2) (2015) 110–125.
25. E.M. Lopez, C.A.C. Coello, Improving the integration of the IGD+ indicator into the selection mechanism of a multi-objective evolutionary algorithm, 2017 IEEE Congress on Evolutionary Computation (CEC) (2017) 2683–2690.
26. O. Schutze, X. Esquivel, A. Lara, C.A.C. Coello, Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 16 (4) (2012) 504–522.
27. D. Brockhoff, T. Wagner, H. Trautmann, R2 indicator-based multiobjective search, Evol. Comput. 23 (3) (2015) 369–395.
28. M.P. Hansen, A. Jaszkiewicz, Evaluating the Quality of Approximations to the Non-Dominated Set, IMM, Department of Mathematical Modelling, Technical University of Denmark, 1998.
29. H. Trautmann, T. Wagner, D. Brockhoff, R2-EMOA: focused multiobjective search using R2-indicator-based selection, in: International Conference on Learning and Intelligent Optimization, Springer, 2013, pp. 70–74.

30. R.H. Gymez, C.A.C. Coello, MOMBI: a new metaheuristic for many-objective optimization based on the R2 indicator, Proceedings of IEEE Congress on Evolutionary Computation (2013) 2488–2495.
31. D.H. Phan, J. Suzuki, R2-BEAN: R2 indicator based evolutionary algorithm for noisy multiobjective optimization, The 2014 Seventh IEEE Symposium on Computational Intelligence for Security and Defense Applications (2014) 1–8.
32. F. Li, J. Liu, S. Tan, X. Yu, R2-MOPSO: a multi-objective particle swarm optimizer based on R2-indicator and decomposition, in: Proceedings of IEEE Congress on Evolutionary Computation, IEEE, 2015, pp. 3148–3155.
33. J.G. Falcyn-Cardona, C.A.C. Coello, A new indicator-based many-objective ant colony optimizer for continuous search spaces, Swarm Intell. 11 (1) (2017) 71–100.
34. Y. Tan, Y. Jiao, H. Li, X. Wang, MOEA/D+ uniform design: a new version of MOEA/D for optimization problems with many objectives, Comput. Oper. Res. 40 (6) (2013) 1648–1660.
35. B. Derbel, D. Brockhoff, A. Liefooghe, S. Verel, On the impact of multiobjective scalarizing functions, in: Parallel Problem Solving from Nature-PPSN XIII, Springer, 2014, pp. 548–558.
36. [36] R.H. Gymez, C.A.C. Coello, Improved metaheuristic based on the R2 indicator for many-objective optimization, in: Proceedings of the Genetic and Evolutionary Computation Conference, ACM, 2015, pp. 679–686.
37. M. Asafuddoula, T. Ray, R. Sarker, A decomposition-based evolutionary algorithm for many objective optimization, IEEE Trans. Evol. Comput. 19 (3) (2015) 445–460.
38. R. Cheng, Y. Jin, M. Olhofer, B. Sendhoff, A reference vector guided evolutionary algorithm for many-objective optimization, IEEE Trans. Evol. Comput. 20 (5) (2016) 773–791.

39. D.H. Phan, J. Suzuki, R2-IBEA: R2 indicator based evolutionary algorithm for multiobjective optimization, in: Proceedings of the IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 1836–1845.
40. K. Miettinen, Nonlinear Multiobjective Optimization, volume 12 of International Series in Operations Research and Management Science, 1999.
41. K. Deb, R.B. Agrawal, Simulated binary crossover for continuous search space, Complex Syst. 9 (3) (1994) 1–15.
42. K. Deb, M. Goyal, A combined genetic adaptive search (GeneAS) for engineering design, Comput. Sci. Inf. 26 (1996) 30–45.
43. J.A. Cornell, Experiments With Mixtures: Designs, Models, and the Analysis of Mixture Data, vol. 895, John Wiley & Sons, 2011.
44. K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable multi-objective optimization test problems, in: Proceedings of the IEEE Congress on Evolutionary Computation, vol. 1, IEEE, 2002, pp. 825–830.
45. S. Huband, P. Hingston, L. Barone, L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. Evol. Comput. 10 (5) (2006) 477–506.
46. Y. Tian, R. Cheng, X. Zhang, Y. Jin, PlatEMO: A MATLAB Platform for Evolutionary Multi-Objective Optimization, 2017, arXiv preprint arXiv:1701.00879.
47. M. Hollander, D.A. Wolfe, E. Chicken, Nonparametric Statistical Methods, John Wiley & Sons, 2013.
48. Y. Wang, B.C. Wang, H.X. Li, G.G. Yen, Incorporating objective function information into the feasibility rule for constrained evolutionary optimization, IEEE Trans. Cybern. 99 (2015) 1–15.
49. R. Cheng, Y. Jin, M. Olhofer, B. Sendhoff, Test problems for large-scale multiobjective and many-objective optimization, IEEE Trans. Cybern. 99 (2016) 1–14.
50. https://www.kaggle.com/c/digit-recognizer/data.

51. https://www.asimovinstitute.org/neural-network-zoo/

52. https://www.asimovinstitute.org/neural-network-zoo-prequel-cells-layers/

53. *Sammut C., Webb J. (Eds.)*. Encyclopedia of Machine Learning. — NY, USA : IBM T. J.Watson Research Center, 2010. — T. 1. — C. 829-838. — ISBN 978-0-387-30768-8.

54. Liang J., Qu B., Mao X., Chen T. (2012) Differential Evolution Based on Fitness Euclidean-Distance Ratio for Multimodal Optimization. In: Huang DS., Gupta P., Zhang X., Premaratne P. (eds) Emerging Intelligent Computing Technology and Applications. ICIC 2012. Communications in Computer and Information Science, vol 304. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-31837-5_72

55. Hancock. P, "A comparison of selection mechanisms", . In Handbook of Evolutionary Computation, Eds . IOP Publishing and Oxford University Press., Bristol, UK, 1997

56. https://github.com/EpistasisLab/tpot

57. https://machinelearningmastery.com/adam-optimization-algorithm-for-deep-learning/

ДОДАТКИ

ДОДАТОК А

Блок-схема алгоритму багатокритеріальної системи оптимізації налаштування нейронних мереж

ДОДАТОК Б

Структурна схема багатокритеріальної системи оптимізації налаштування нейронних мереж

ДОДАТОК В

Блок-схема основних етапів будь-якого генетичного алгоритму

ДОДАТОК Г

Перелік ШНМ на даний час

ДОДАТОК Д

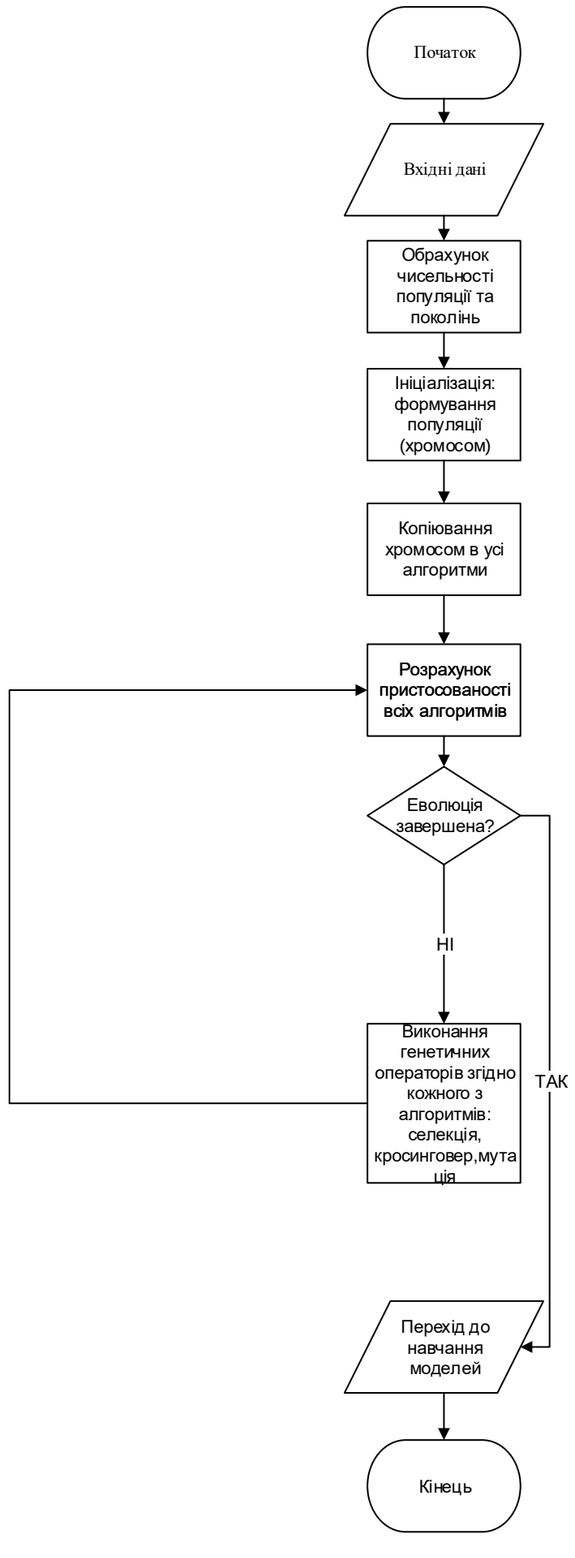
Найкращі результати різних нейронних мереж на MNIST наборі даних

ДОДАТОК Е

Результати багатокритеріальної системи оптимізації налаштування нейронних мереж

Додаток Є
Результат перевірки роботи на співпадіння

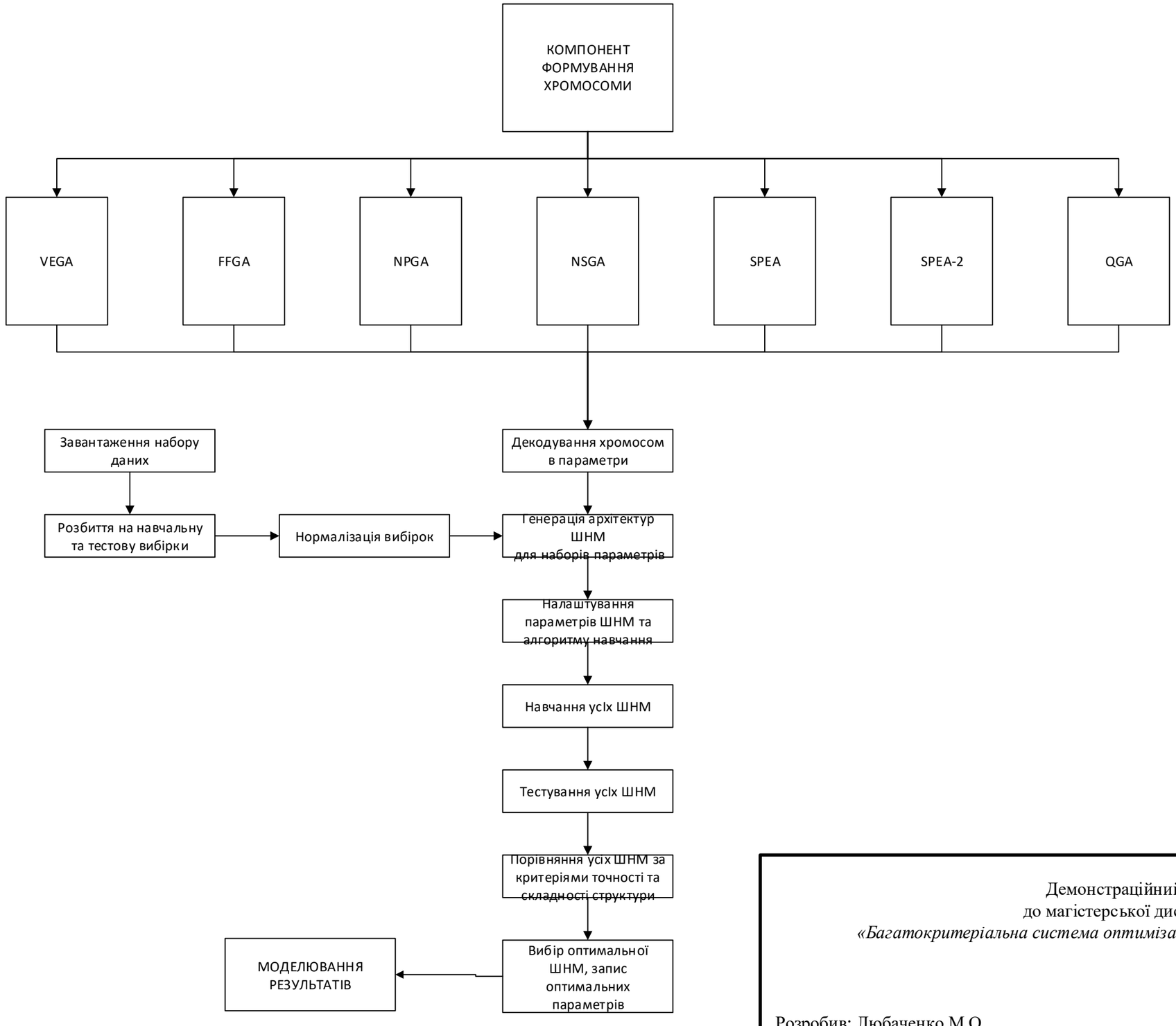
Блок-схема алгоритму багатокритеріальної системи оптимізації налаштування нейронних мереж



Демонстраційний плакат № 1
до магістерської дисертації на тему
«Багатокритеріальна система оптимізації налаштування нейронних мереж»

Розробив: Любаченко М.О.
Прийняла: Чумаченко О.І.

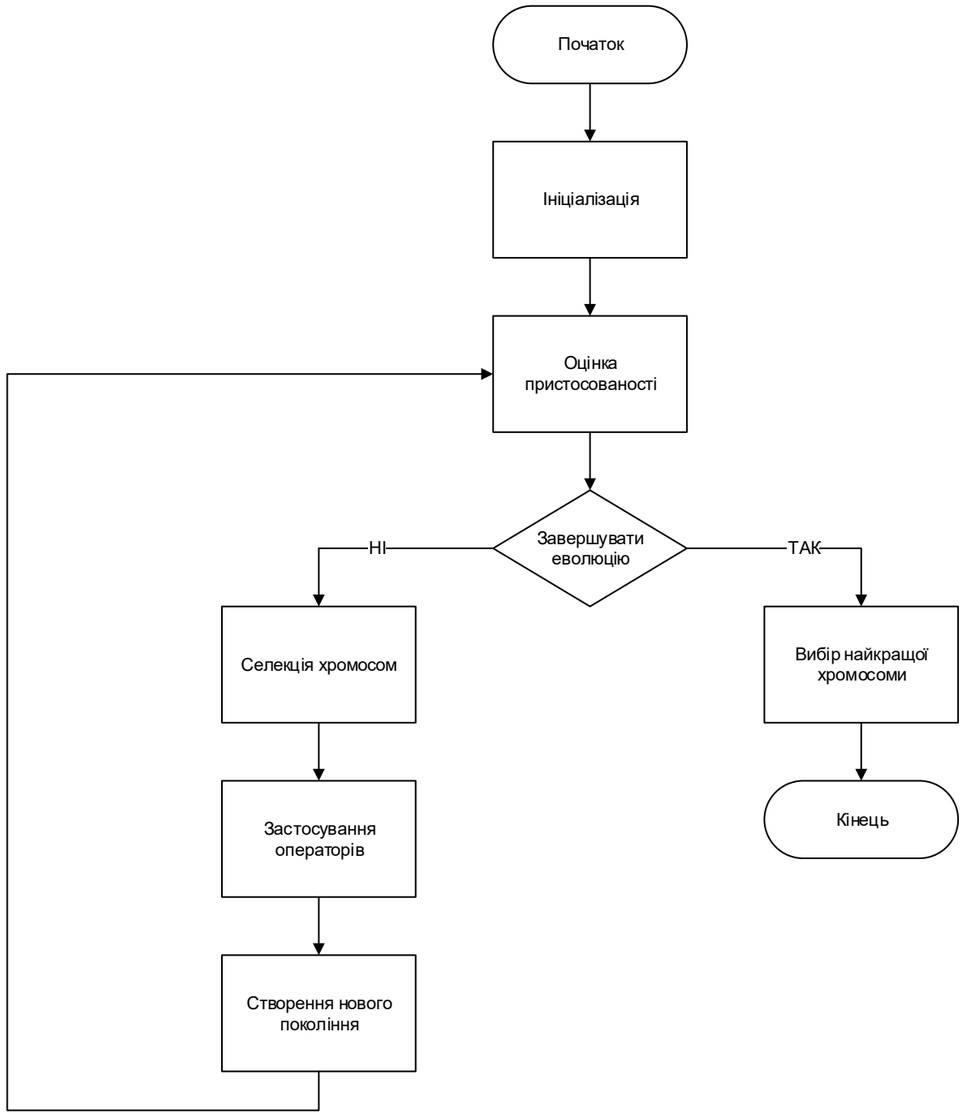
Структурна схема багатокритеріальної системи оптимізації налаштування нейронних мереж



Демонстраційний плакат № 1
до магістерської дисертації на тему
«Багатокритеріальна система оптимізації налаштування нейронних мереж»

Розробив: Любаченко М.О.
Прийняла: Чумаченко О.І.

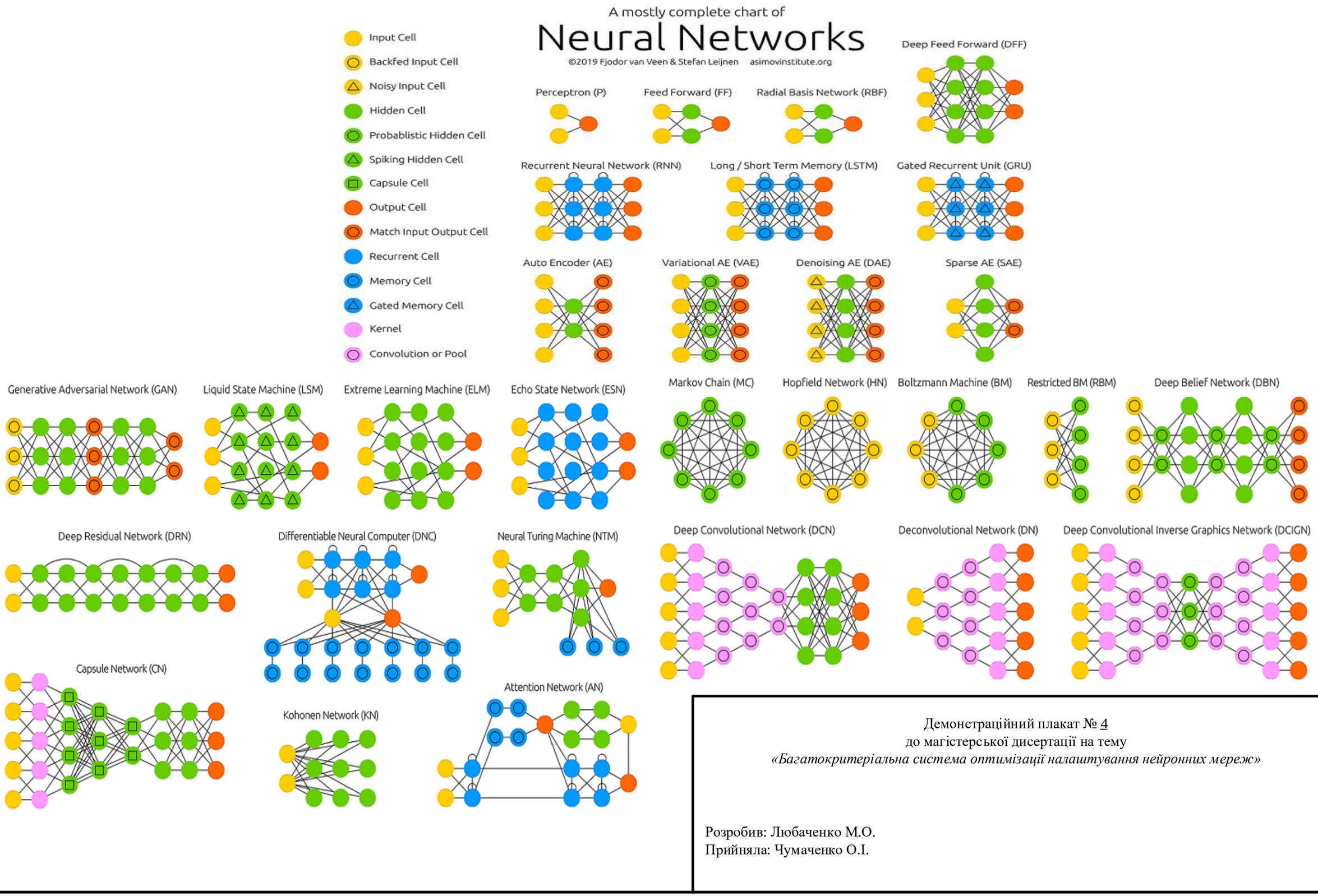
Блок-схема основних етапів будь-якого генетичного алгоритму



Демонстраційний плакат № 3
до магістерської дисертації на тему
«Багатокритеріальна система оптимізації налаштування нейронних мереж»

Розробив: Любаченко М.О.
Прийняла: Чумаченко О.І.

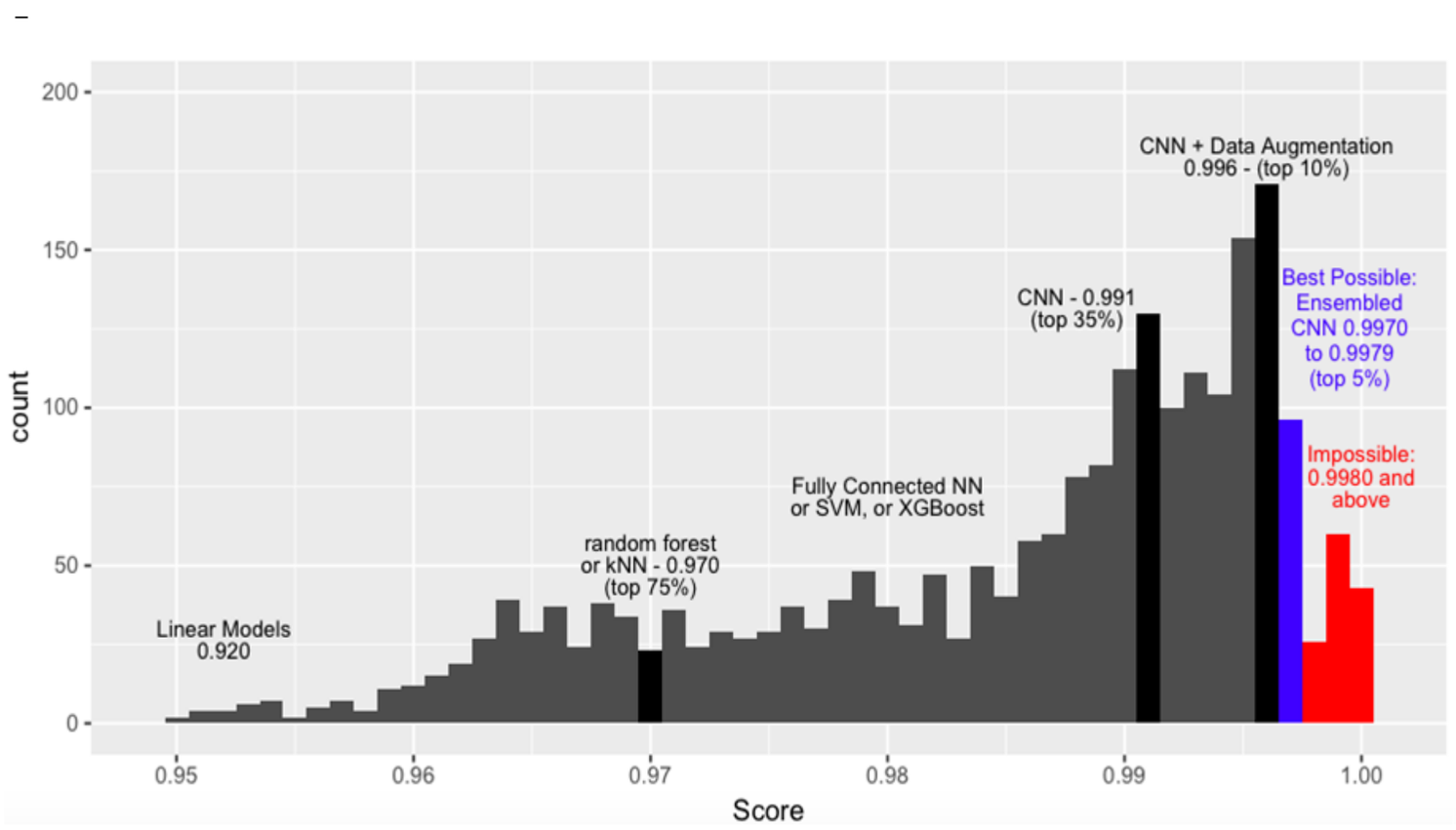
Перелік ШНМ на даний час



Демонстраційний плакат № 4
до магістерської дисертації на тему
«Багатокритеріальна система оптимізації налаштування нейронних мереж»

Розробив: Любаченко М.О.
Прийняла: Чумаченко О.І.

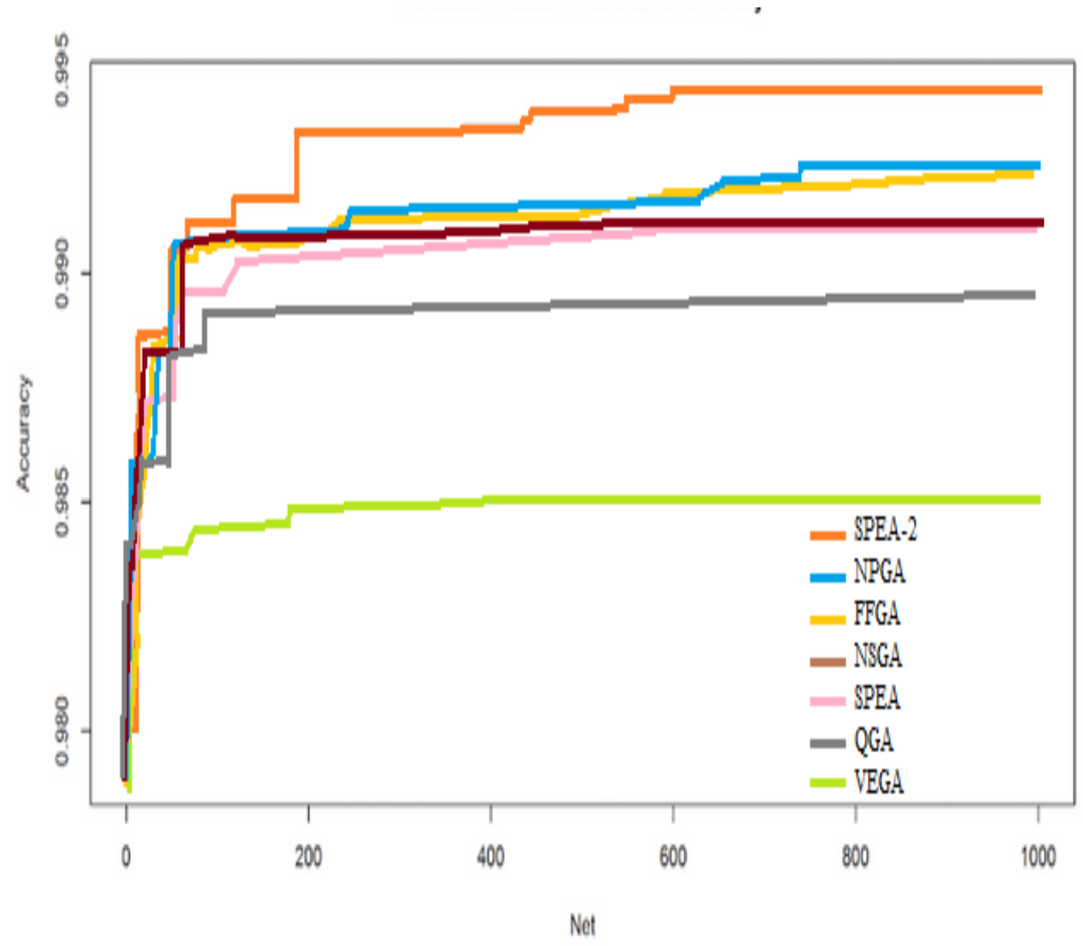
Найкращі результати різних нейронних мереж на MNIST наборі даних



Демонстраційний плакат № 5
до магістерської дисертації на тему
«Багатокритеріальна система оптимізації налаштування нейронних мереж»

Розробив: Любаченко М.О.
Прийняла: Чумаченко О.І.

Результати багатокритеріальної системи оптимізації налаштування нейронних мереж



Демонстраційний плакат № 6
до магістерської дисертації на тему
«Багатокритеріальна система оптимізації налаштування нейронних мереж»

Розробив: Любаченко М.О.
Прийняла: Чумаченко О.І.

UNICHECK

ІК-91мп Любаченко М.О. (Співпадіння)

Завантажено: 12/10/2020 | **Перевірено:** 12/10/2020

Matches Цитата Використані джерела Заміна символів

1.4% Matches

0% Цитати

0% Використані дж

Matches

Веб джерела 55

1	**www.cs.bham.ac.uk** https://www.cs.bham.ac.uk/~chengr/Preprint/TSEA.pdf	0.47%
2	**ukrbukva.net** https://ukrbukva.net/page,10,89472-Ispol-zovanie-geneticheskih-algoritmov-dlya-optimizacii-bazy-pravil.html	0.21%
3	**ejournal.uksw.edu** https://ejournal.uksw.edu/scholaria/article/download/2620/1372/	0.18%
4	**uabs.sumdu.edu.ua** https://uabs.sumdu.edu.ua/images/stories/docs/2133/Kuzmenko.pdf	0.18%
5	**www.lib.nau.edu.ua** http://www.lib.nau.edu.ua/Journals/frmDoc.aspx?param=124	0.18%
6	**www.pitt.edu** http://www.pitt.edu/~kmram/0132/lectures/registers+counters.pdf	0.18%
7	**aaltodoc.aalto.fi** https://aaltodoc.aalto.fi/bitstream/handle/123456789/31579/master_Khamehchi_Sina_2018.pdf?sequence=1	0.18%
8	**asktom.oracle.com** https://asktom.oracle.com/pls/apex/f?p=100:11:0::::P11_QUESTION_ID:275215756923	0.18%
9	**emasf2.webcindario.com** https://emasf2.webcindario.com/EmasF_53.pdf	0.18%
10	**uk.wikipedia.org** https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D1%96%D1%8F_%D0%BE%D0%BF%D1%82%D0%B8…	0.18%